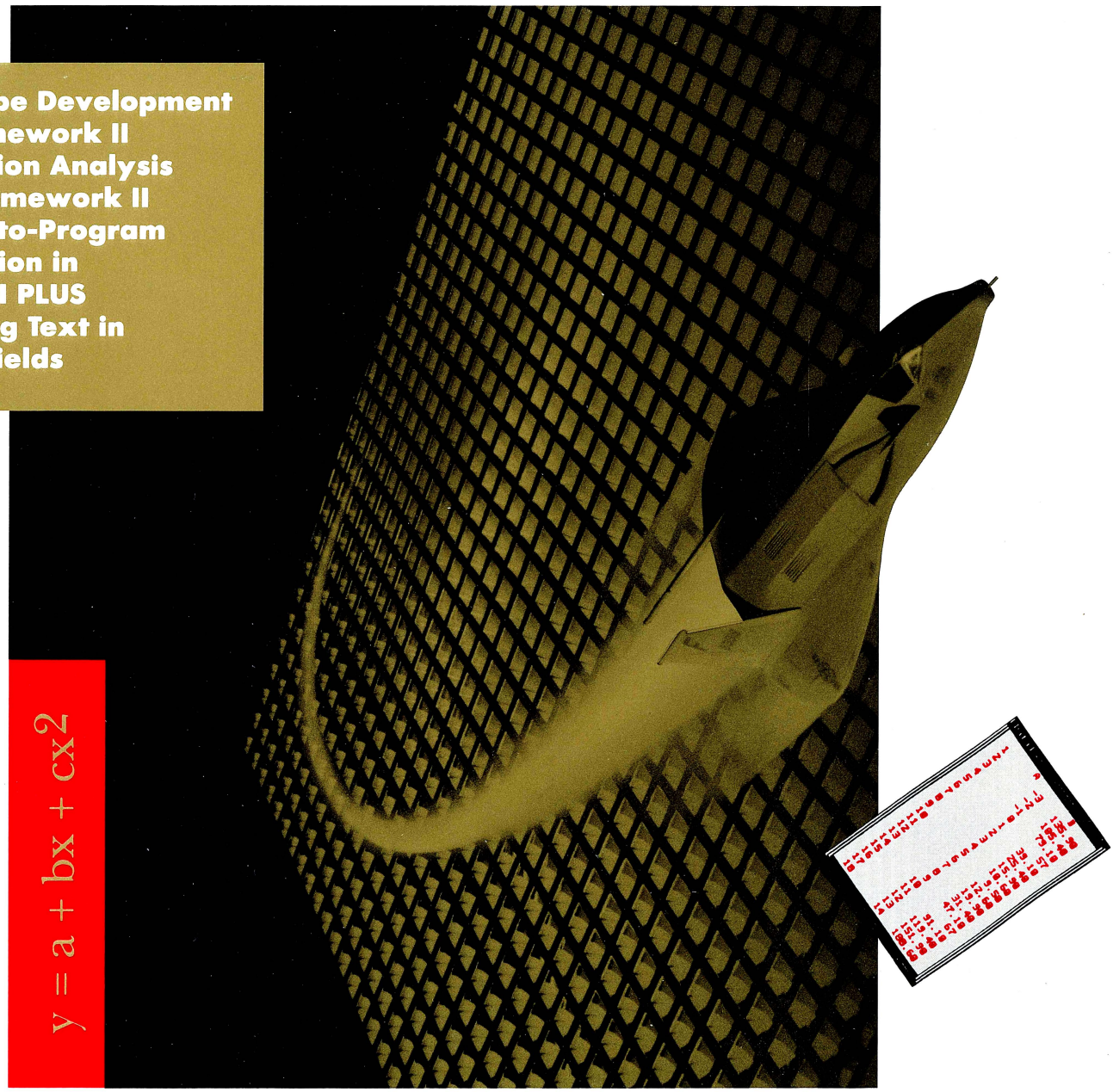
TechNotes

ASHTON-TATE'S TECHNICAL JOURNAL
FEATURING dBASE AND FRAMEWORK

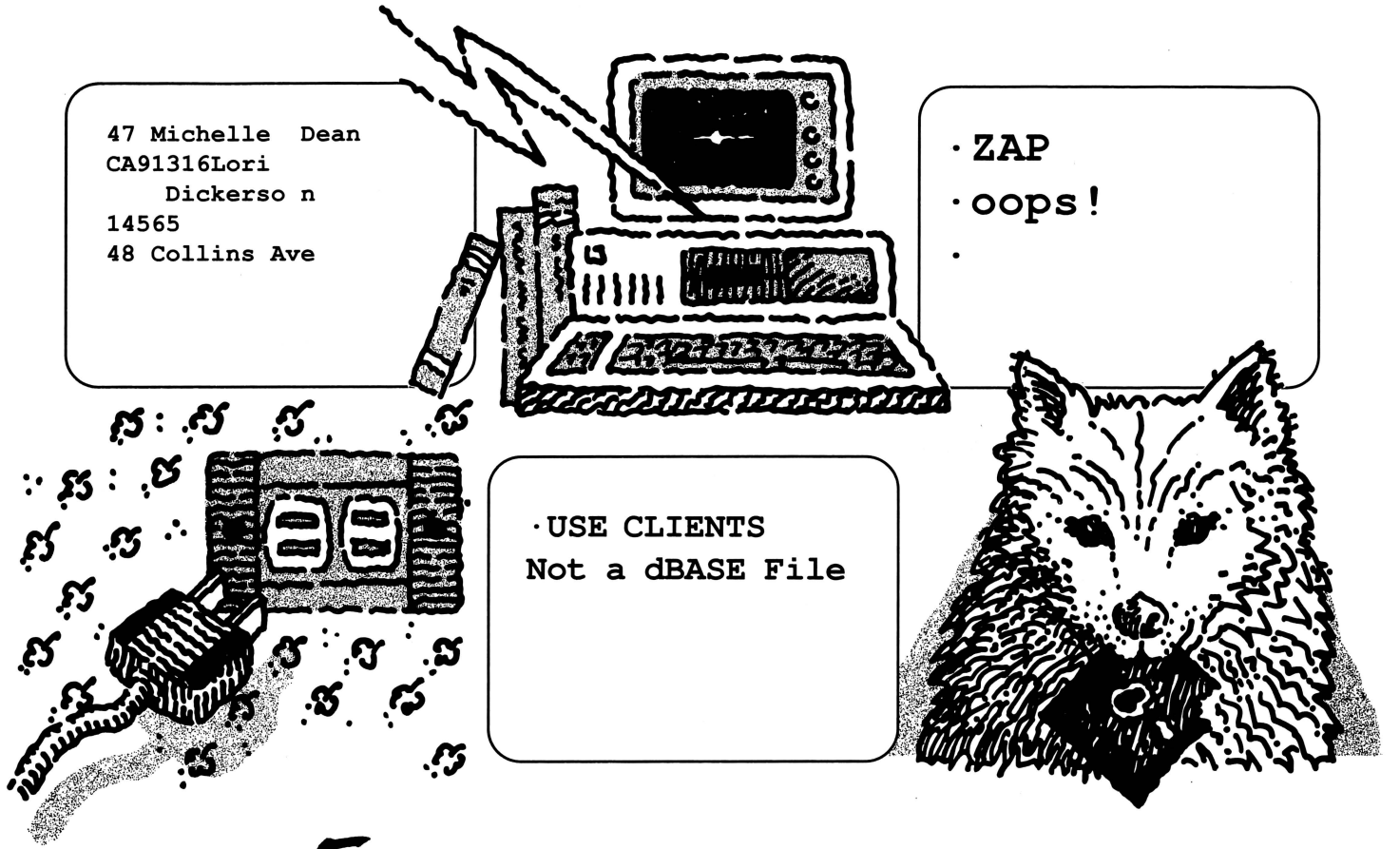
JANUARY 1988
$6.00

FEB 3

- **Prototype Development for Framework II**
- **Regression Analysis with Framework II**
- **Format-to-Program Conversion in dBASE III PLUS**
- **Detecting Text in Memo Fields**



ASHTON·TATE



5 ~~6~~ Reasons To Buy *dBASE® File Recovery*

We can't help you if the dog eats your data.

But *dBASE File Recovery* does give you the ability to recover damaged data in dBASE III® database files, including:

- "Shifted" data
- Corrupted data
- Missing or corrupted dBASE file headers
- Erased or ZAPped dBASE files
- Data from a BACKUP that you can't RESTORE

Don't let a damaged database file ruin your day!

With the programs on the *dBASE File Recovery* disk, you can diagnose and repair data damage . As long as the data still exists on the disk, *dBASE File Recovery* can help get it back!

It's dBASE data insurance for only $49.95!

dBASE File Recovery is available from the Support Products Center at Ashton-Tate, 20101 Hamilton Avenue, Torrance, CA 90502-1319.

Call 213/538-7482 to order.

TechNotes

JANUARY 1988

dBASE III PLUS

Format to Program Conversion 5

Step-by-step instructions for translating your format screens into printing programs.

Detecting Text in Memo Fields 10

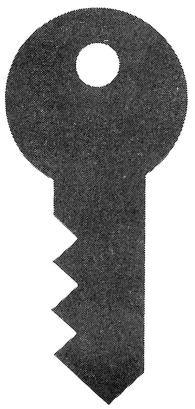
Want to know if there's text in a memo field? Ismemo.BIN lets you know and you don't have to capture the memo text in another file.

A Soft Record-Locking Subroutine 22

The Programmer's Casebook tackles network programming with a subroutine to use with the dBASE Administrator.

Printing with Special Effects 28

SetHelpOn, our new column for novice users, provides a tutorial on fancy printing.
by Naoma Stow



FRAMEWORK II

Regression Analysis 34

An implementation of the well-known statistical procedure examines relationships between variables stored in Framework II spreadsheets and databases.

Function Prototyping 44

This tool helps you construct and test complicated FRED expressions on the desktop before moving them into your programs.

Printer Driver Update 51

A complete table of the printer drivers currently available for Framework II.

DEPARTMENTS

dBASE III PLUS Usage Tips 31
Framework II Usage Tips 57
RapidFile Usage Tips 62
Anomalies 67
Corrigendum 70
On-Line Services 71
Letters 75
Softstrips™ 77
Classifieds 80

ASHTON-TATE SOFTWARE SUPPORT PUBLICATION

TechNotes

Publisher
Ashton-Tate
Support Products Center
Richard Goepel, Manager

Editor-in-Chief
Brenda Johnson-Grau

Editors
James Bradbury
Ken Getz (Emeritus)
Charles Litzell

Production Editor
Tom Pasley

Editorial Assistant
Siobhan Brownlow

Technical Staff
Larry Abare, Rozita Afar, Richard Augelli, Rodney Barbati, Richard Bunter, Mike Carlisle, James Chuang,Fred Crownover, Rick Fillman, Peter Gold, Mina Gussin, Scott Haynes, J. Mike Hedblom, Clint Helton, Errol Jackson, Matt Johnson, Gail Kulikowsky, Steve Kurasch, Chris LoVasco, Jeffrey McCrimon, Jerry Mendes, Vincent Mendillo, Jeffrey Moore, Joseph Perret, John Regen, Cathy Shikatani, Joseph Stolz, Naoma Stow, Philip Talsky, Mitra Toossi, Hiram Vega, Ralph Wakefield, Mark Ware,Roger Wegehoft, Quinn Wildman,Tom Woodward

Contributing Editors
Paul Bukauskas (International), Liz Burke (MultiMate), Steve Silverwood (On-Line Services)

Production Staff
Gary Ashcavai, D.A. Gary Bare, Tim Dunne, Darrell Evers, Patty Pesek, Ron Powell, Ann O. Taono

Product Development
Forefront Development Center: Marilyn McMahon, Greg Stikeleather Glendale Development Center

DISCLAIMER NOTICE

The information contained in this publication is designed to help you develop and improve your programming skills and to assist you in using Ashton-Tate products. Ashton-Tate assumes no responsibility whatsoever for the uses made of any of the software code in this issue, whether modified or unmodified.

If you discover what you believe is an anomaly (as defined under **Anomalies**) not presented in this issue or others, please send a detailed description of the possible anomaly to:

Ashton-Tate Software Support Center
20101 Hamilton Avenue
Torrance, CA 90502-1319

If the reported anomaly can be reproduced by our technicians, it may be added to a future issue of ***TechNotes.***

The policies that are described in ***TechNotes*** apply only to the United States (and possessions), Canada, and Mexico. Residents of other foreign countries should contact a local Ashton-Tate representative in their country for policy information and technical support.

Ashton-Tate and the authors will not be liable for special, incidental, consequential, indirect or other similar damages, even if we have been advised of the possibility of such damages. This means we are not responsible for damages or costs incurred as a result of loss of time, loss of data, loss of profits or revenues, or loss of use of the software. In addition, we are not responsible for damages or costs incurred in connection with obtaining substitute software, claims by others, inconvenience or similar costs.

No warranties, either expressed or implied, are made regarding the contents of this publication, its merchantability or fitness for any particular purpose, all of which are hereby expressly disclaimed. Note, however, that some states do not allow the exclusion or limitation of warranties or of direct, incidental or consequential damages, so these warranty limitations may not apply to you.

In no event will our liability for any damages to you or any other person ever exceed the price paid for your subscription to ***TechNotes***, regardless of any form of the claim.

Additional statements by agents, employees, distributors and dealers of Ashton-Tate do not constitute warranties by Ashton-Tate and do not bind Ashton-Tate.

TechNotes (ISSN 0883-6493) is an Ashton-Tate Support Products Center publication containing technical reference information on Ashton-Tate products.

Entire contents Copyright © Ashton-Tate Corporation 1987. All rights reserved.

Commercial reproduction of the material appearing in ***TechNotes*** is forbidden without prior written permission. Direct all requests for reproduction permission to Ashton-Tate Support Services, 20101 Hamilton Avenue, Torrance, CA 90502-1319, ATTN: Brenda Johnson-Grau.

TechNotes is published monthly at a subscription price of $50.00 for one year. Add $15 for Canada and Mexico and add $25 for overseas. ***TechNotes*** is published by Ashton-Tate, 20101 Hamilton Avenue, Torrance, CA 90502-1319.

POSTMASTER: Send subscription inquiries and address changes to ***TechNotes*** Subscription Service, P.O. Box 3729, Escondido, CA 92025-0929.

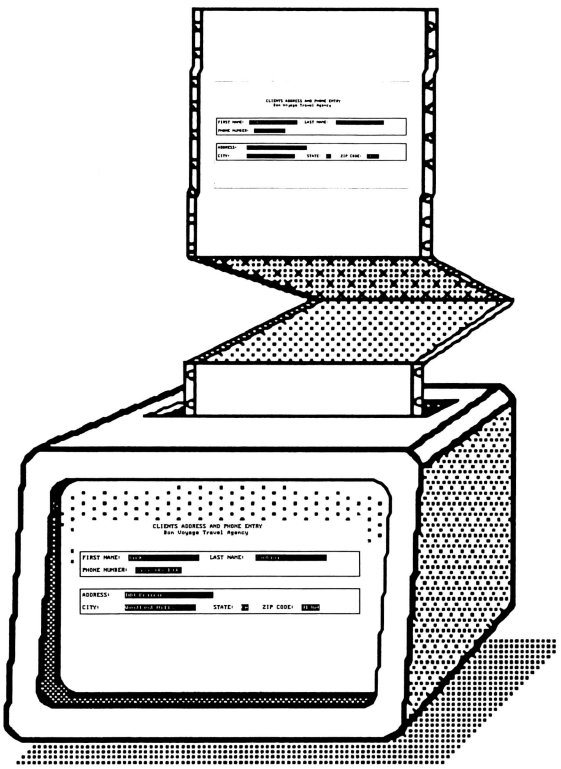
Subscription Inquiries: Write ***TechNotes*** Customer Service, P.O. Box 3729, Escondido, CA 92025-0929. Call 800/828-2515 or 800/331-4164. From outside the U.S. call: 619/485-6535. When writing or calling, be sure that you have your current mailing label available. Your expiration date is listed on the far right of your mailing label.

dBASE, dBASE II, dBASE III, Ashton-Tate and Framework are registered trademarks and RapidFile, dBASE III PLUS, Framework II, RunTime, and FRED are trademarks of Ashton-Tate Corporation. Some of the trademarks of other companies mentioned in this issue appear for identification purposes only. MultiMate is a registered trademark and MultiMate Advantage, MultiMate GraphLink and the MultiMate Business Advantage Keyboard are trademarks of MultiMate International Corporation, an Ashton-Tate company. On-File is a trademark under license from Facts on File, Inc.

The Ashton-Tate Software Support Center's hours are 6:00 a.m. to 4:30 p.m., PST, Monday through Friday, except on Thursday when we close at 4:00 p.m. To reach our technicians for technical support, call (213) 329-0086. Be sure to have your software serial number ready for our operators.

Format-to-Program Conversion

By J. Mike Hedblom and Chuck Litzell



The article, "Converting a Format File to a Simple Form Report," which appeared in the July, 1986 ***TechNotes,*** addresses one of the most frequent questions from new dBASE III PLUS users: "Now that I've created a format screen, how do I print it?"

The article outlined the steps needed to create a printing program from the .FMT screen generated by the CREATE/MODIFY SCREEN command. In this article, we'll repeat these initial conversion steps, discuss obstacles you may encounter in the process, and provide methods of enhancing your program to produce more customized reports.

CREATE/MODIFY SCREEN is a simple yet powerful utility for designing data-entry screens, since it allows you to format the screen just the way you want it. Format screens are intended for data entry, but with a little effort they can be modified to produce printed reports. This technique is also a good introduction to programming in dBASE III PLUS since it introduces the commands that perform formatted input and output, and the DO WHILE loop, a program-only command found in nearly all dBASE III PLUS programs.

When you design a screen format with CREATE/MODIFY SCREEN, two files are produced. The files have the same basename but different extensions. The screen design is stored in a file with an .SCR extension. This file is accessed only by the MODIFY SCREEN command. The design is also translated into the English-like dBASE III PLUS commands, and stored in a text file with an .FMT extension.

A format file (.FMT) is a special form of a dBASE III PLUS program file. When the command

```
SET FORMAT TO <screen name>
```

is issued, dBASE loads the contents of <screen name>.FMT into memory.

```
@  4, 21  SAY "CLIENTS ADDRESS AND PHONE ENTRY"
@  5, 24  SAY "Bon Voyage Travel Agency"
@  9,  1  SAY "FIRST NAME:"
@  9, 14  GET  CLIENTS->FIRSTNAME
@  9, 37  SAY "LAST NAME:"
@  9, 50  GET  CLIENTS->LASTNAME
@ 11,  1  SAY "PHONE NUMBER:"
@ 11, 16  GET  CLIENTS->PHONE
@ 15,  1  SAY "ADDRESS:"
@ 15, 13  GET  CLIENTS->ADDRESS
@ 17,  1  SAY "CITY:"
@ 17, 13  GET  CLIENTS->CITY
@ 17, 38  SAY "STATE:"
@ 17, 46  GET  CLIENTS->STATE
@ 17, 52  SAY "ZIP CODE:"
@ 17, 63  GET  CLIENTS->ZIPCODE
@  8,  0  TO 12, 79
@ 14,  0  TO 18, 79
```

Figure 1: Clients.FMT

Thereafter, any full-screen commands, such as EDIT, APPEND, or READ, activate the format screen. The only dBASE III PLUS commands in the format screen are those that control data input and output in full-screen mode:

```
@...SAY     && Display data
@...GET     && Input data
@...TO      && Draw lines or boxes
READ        && Activate screen page
```

Figure 1 is a listing of Clients.FMT, a format file found on the dBASE III PLUS "Sample Programs & Utilities" diskette. Clients.FMT uses three commands: @...SAY, @...GET, and @...TO. In all three cases, row and column coordinates are used. dBASE numbers rows and columns beginning at 0. The upper left-hand corner of the screen is row 0, column 0. The bottom right-hand corner is row 24, column 79. For example,

```
@ 4, 21 SAY "CLIENTS ADDRESS AND PHONE ENTRY"
```

tells dBASE to move the cursor to line 4, column 21 and display (SAY), "CLIENTS ADDRESS AND PHONE ENTRY."

```
@ 9, 14 GET CLIENTS->FIRSTNAME
```

tells dBASE to input (GET) into the Firstname field of the current record in Clients.DBF, beginning at row 9, column 14.

```
                CLIENTS ADDRESS AND PHONE ENTRY
                  Bon Voyage Travel Agency

FIRST NAME:  Rick                 LAST NAME:  Zambini
PHONE NUMBER:  (555)345-1945

ADDRESS:  108 Prairie
CITY:  Woodland Hills        STATE:  CA   ZIP CODE:  91364
```

Figure 2: Format screen dump.

The line- and box-drawing commands include two sets of row and column coordinates.

```
@  8,  0  TO 12, 79
```

draws a box with corners at row 8, column 0 and row 12, column 79.

Figure 2 shows a screen formatted with Clients.FMT.

You should notice a few points about the way Clients.FMT is organized. Since it is a one-page screen, no READ commands are needed. With the exception of the two box-drawing commands at the end, the row numbers are in ascending order, from 4 through 17. When there is more than one item on a line, the column numbers are in ascending order.

Format screens produced by dBASE always follow these rules. The SAY and GET commands are stored in ascending row/column order, and line and box drawing commands always appear at the end of each screen page. If Clients.FMT had a second page, dBASE would have added a READ command immediately after the existing lines, followed by the SAY and GET commands for the second page, and then any line or box drawing commands.

Sending Formatted Output to the Printer

@...SAY commands may be redirected to the printer by issuing the SET DEVICE TO PRINT command. Printer redirection is canceled with the SET DEVICE TO SCREEN command. dBASE maintains internal row and column counters for the printer. When output has been redirected with the SET DEVICE command, dBASE uses its counters to determine how many linefeeds and spaces must be sent to move the printer's carriage to the coordinates specified.

The order in which @...SAY commands are issued is critical when DEVICE has been set to PRINT. For example, these commands print on two separate sheets of paper:

```
@ 5, 0 SAY "Line 5"
@ 4, 0 SAY "Line 4"
```

After "Line 5" has been printed, the printer's carriage is positioned on row 5, column 6. In order to get to the previous row the printer would need to perform a reverse linefeed. Many printers do not have this capability. In any case, dBASE always moves paper *forward.* A request to print at a row lower than the current row is understood as a request to begin a new page, and a form-feed character (CHR(12)) is sent to the printer.

Since CREATE/MODIFY FORMAT always puts @...SAY commands in ascending order by row number, a problem exists only if the format screen has more than one page. If you want to print multiple screen pages on a single sheet of paper, you will need to adjust the row numbers to prevent dBASE III PLUS from sending formfeed characters between screen pages.

```
@ PROW()+1, 21  SAY "CLIENTS ADDRESS AND PHONE ENTRY"
@ PROW()+1, 24  SAY "Bon Voyage Travel Agency"
@ PROW()+4,  1  SAY "FIRST NAME:"
@ PROW()   , 14  GET  CLIENTS->FIRSTNAME
@ PROW()   , 37  SAY "LAST NAME:"
@ PROW()   , 50  GET  CLIENTS->LASTNAME
@ PROW()+2,  1  SAY "PHONE NUMBER:"
@ PROW()   , 16  GET  CLIENTS->PHONE
@ PROW()+4,  1  SAY "ADDRESS:"
@ PROW()   , 13  GET  CLIENTS->ADDRESS
@ PROW()+2,  1  SAY "CITY:"
@ PROW()   , 13  GET  CLIENTS->CITY
@ PROW()   , 38  SAY "STATE:"
@ PROW()   , 46  GET  CLIENTS->STATE
@ PROW()   , 52  SAY "ZIP CODE:"
@ PROW()   , 63  GET  CLIENTS->ZIPCODE
```

Figure 3: Clients.FMT Rewritten with PROW()

Conversion Process

These are the steps required to redirect a format screen to the printer.

1. Copy your format file to a program file with the dBASE command:

```
COPY FILE <screen name>.FMT TO <screen name>.PRG
```

2. Edit the new program by typing

```
MODIFY COMMAND <screen name>
```

at the dot prompt.

3. Replace every occurrence of "GET" with "SAY". For example,

```
@ 10, 05 GET MASTER->LASTNAME
```

becomes

```
@ 10, 05 SAY MASTER->LASTNAME
```

4. Add the beginning code sequence by moving the cursor to the first row, pressing **Ins**, and inserting the lines which follow. Replace "<database file>" with the name of your database, and "<index file>" with the name of the index you want to use. You do not need to type the .DBF or .NDX filename extensions. (Lines beginning with "*" are comments and need not be entered.)

```
SET TALK OFF
* ---The next two lines necessary only if you're
* ---printing memo fields. <n> is the width,
* ---in columns, of printed memos.
SET MEMOWIDTH TO <n>
SET PRINT ON
SET DEVICE TO PRINT
USE <database file> INDEX <index file>
GO TOP
DO WHILE .NOT. EOF()
```

5. Add the ending code sequence by moving the cursor to the last line of the program, pressing **Return** to advance to the next available line, and entering the following lines of code:

```
SKIP
ENDDO
EJECT
* ---SET PRINT OFF is necessray only
* ---if you're printing memos
SET PRINT OFF
SET DEVICE TO SCREEN
CLOSE DATABASES
SET TALK ON
RETURN
```

6. Press **Ctrl-End** to save your work and return to the dot prompt.

Running the Report

After these changes have been made, your dBASE III PLUS program will print each record in the database, one per page. To run the report, turn on your printer and enter

```
DO <screen name>
```

at the dot prompt. This command tells dBASE to load and execute the commands in your new program file.

Special Considerations

Printing memo fields, multi-screen format files, boxes, and printing more than one record per page require some special handling.

Printing Memo Fields

Memo fields require special attention when converting .FMT files to .PRG files. @...SAY commands are redirected to the printer using the SET DEVICE TO PRINT command. Memo text, however, cannot be displayed or printed with an @...SAY command. Instead, you must print the memo with ? or ??.

Within the new .PRG file, replace lines like

```
@ xx, yy GET <memo field>
```

with

```
@ xx, 00 SAY ""
?? SPACE(yy - 1), <memo field>
```

If col is 00, simply enter

```
@ row, 00 SAY ""
?? <memo field>
```

Note, too, that you can not tell how many printed lines a memo field will take prior to printing it. For this reason, make sure your memo field(s) are the last fields to be printed. Printing fields with @...SAY commands after a memo field may cause the program to eject pages at undesirable times.

Multi-Screen Formats

A .FMT file may have up to ten *pages*. When used for data entry the **PgUp** and **PgDn** keys move between pages of a screen. Each time you move to another screen page, dBASE clears the current screen and draws the new page.

A READ command is the signal for a new page in a format file. The row numbers following the READ command are reset to begin at the top of the new screen. If you don't change them when converting a multi-page .FMT file, each page of the .FMT file will begin on a new sheet of paper. To prevent this you need to do two things. First, remove all

READ commands from your .PRG file. Then, add 25 to the row numbers of items appearing on even-numbered pages. Your program will then print two screen pages on each sheet of paper.

Boxes

If you have used boxes or lines on your screen, dBASE places the commands to draw them following all of the @...SAY and @...GET commands. Lines and boxes come in two varieties: single and double. The drawing commands are

```
@ row1, col1 TO row2, col2          && Single
@ row1, col1 TO row2, col2 DOUBLE   && Double
```

The row and column coordinates specify diagonal corners of the box to be drawn. If row1 is the same as row2, a horizontal line is drawn. If col1 is the same as col2, a vertical line is drawn.

SET DEVICE TO PRINT does not affect the drawing of lines or boxes. While the remainder of your format screen is printing on the printer, lines and boxes will continue to be drawn on the screen. Therefore, you will want to remove these commands from your .PRG file during conversion.

Printing More than One Record per Page

If your format screen takes only a few printed lines, it may be a waste of paper to print just one record per page. To print multiple records per page, you must modify your new program again.

The challenge is to change the @...SAY commands in such a way that the row numbers increase not only within a record but also between records. For example, if each record requires ten printed lines, you could fit six records on a letter-size sheet. The first record would begin on line 1, the second on line 11, the third on line 21, and so on. Your format screen, however, was designed to begin each record at line 1.

What you must do is replace the "hard coded" row numbers with an "indirect reference" to a row number. There are two ways to approach this. One way is to establish a memory variable which will keep track of the row number. The program, then, must keep the memory variable up-to-date.

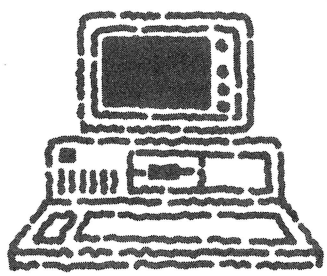
Another way is to use dBASE III PLUS's built-in PROW() function. PROW() is a numeric function, returning the current row number on the printer. (Remember dBASE's internal row and column counters?).

There is a small trick to using PROW() to print multiple records per page. PROW() returns the *current* row number. Certainly you don't want to print an entire report on the current row, so you need a way to increment PROW() when you need to start a new line. Since dBASE updates PROW() automatically all you need to do is print something on the next line.

Figure 3 is Clients.FMT rewritten to use PROW() rather than an absolute row number. Note that each time the row number changes in figure 2, a constant is added to PROW() in figure 3; otherwise, the row number is simply replaced with PROW().

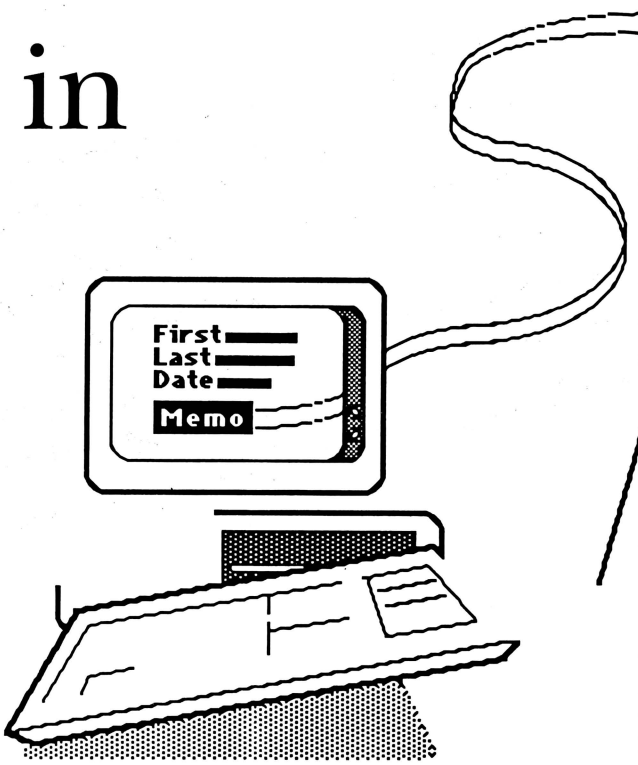
Future Directions

The techniques discussed will help you to convert your format screens to printing programs. From here, it is not a difficult task to add page headings and page numbers to your report programs. Adding totals and subtotals is much more challenging but would produce a report program as capable as the built-in report generator with the flexible layout of the format screen.



Detecting Text in Memo Fields

Ismemo.BIN searches a particular record in your dBASE III PLUS database file looking for an attached memo field.

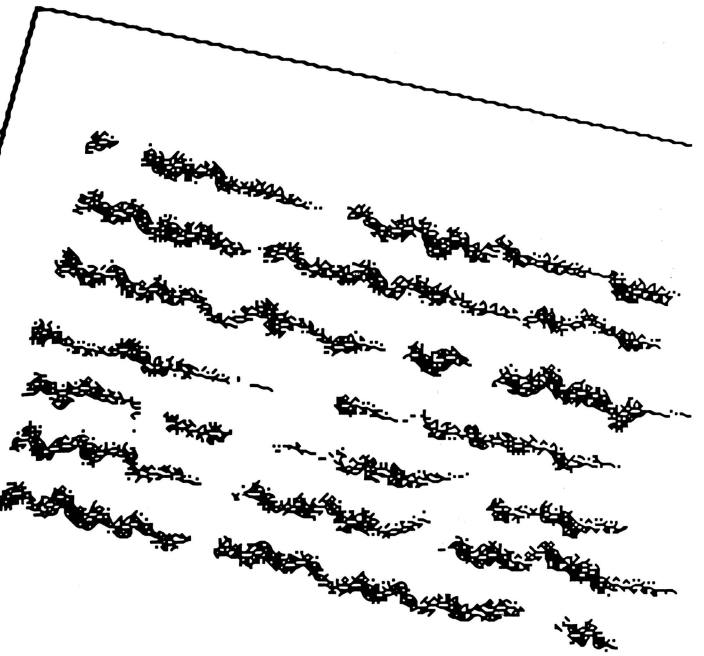


By Kenneth N. Getz and Chuck Litzell

In November of 1987, we published "Memo to Character Conversion in dBASE III PLUS," a program with several accompanying utilities that operated on memo fields by moving them to character fields within a database file. The programs made it possible to search memo text, test for the existence of a memo, or determine the number of text lines in a memo.

One of the caveats mentioned in the article was that moving memo fields to a database file is time-consuming and requires a lot of space on your disk. Because the memo field data must first be output into a text file and then APPENDed into a database, you needed room on your disk to hold three copies of the memos at once. For some applications, that degree of file manipulation is unavoidable, but what if all you want to know is whether there is text in a memo field? There is a better way.

Ismemo.BIN is a binary file that tests for the existence of a memo field for a specific record. It directly reads the database file to test whether dBASE has assigned memo text to a memo field.



A little background on the organization of memo fields will help you understand how Ismemo works. In a database file (.DBF), dBASE III PLUS reserves ten characters for memo fields. No actual text is stored in the database file. Instead, dBASE stores the text in a separate text file that has the same basename as the database file and has a .DBT extension. Thus Clients.DBT would be the memo text file for the database file Clients.DBF.

Within the .DBT file, text is stored in 512-byte blocks. If a memo requires more than 512 characters, it uses additional 512-byte blocks—as many as necessary. The ten-character field in the .DBF file contains a pointer to the memo's first block, stored as a string of characters. If a memo begins in block 567, then the field in the database file contains "567", right-justified. If no memo text exists for a record, then the field contains ten spaces.

Ismemo is CALLed with a memory variable that contains three pieces of information: the offset of the memo field within a record, the record number to be tested, and the name of the database file. Ismemo uses this information to open the database, determine the offset from the beginning of the file to the memo field to be tested, and look there for non-blank characters. If Ismemo finds any non-blank characters, then a memo field exists.

Usage

Ismemo is CALLed WITH a character memory variable, containing three pieces of information.

```
memvar = STR(<memo offset>, 4) +;
       STR(<record number>, 10) +;
       <database filename>
CALL Ismemo WITH memvar
```

The first four bytes are the offset of the memo field within the database record, in character format. The offset is the combined width of all fields preceding the memo field.

Bytes 5 through 14 of the memory variable contain the character representation of the record number that is to be tested. The name of the database file, including the extension, begins at byte 15. If the database is not in the current directory, you should include the full path so that Ismemo can locate the file.

If no error occurs, the first character in the memory variable will be a space (CHR(32)), and the second character will be either "Y" or "N", depending on whether a memo field was found.

If an error occurs, Ismemo places one of the codes shown in Table 1 in the first byte of the memory variable.

The first five of these, CHR(2) through CHR(6), are returned by DOS through calls Ismemo makes to the DOS file services. They generally indicate difficulty in locating or opening the database file. The last four errors are returned by Ismemo and indicate a problem with the parameters passed. If any value other than CHR(2) through CHR(10), or CHR(32) occurs in the first byte of the memory variable, then an unspecified file error has occurred.

Calculating the Memo Field Offset

To calculate the offset of the memo field in the database, you need to add together the widths of all the fields preceding the target memo field. For instance, if a memo field is preceded by two character fields, each 10 characters in length, the offset to the memo field is 20.

You might want to use Ismemo in a program where the database containing the memo field is not known until the program is run. You must either have the user enter the offset or calculate the offset to the memo field on the fly.

Error Number	Meaning
2	File not found.
3	Path not found or file doesn't exist.
4	No file handle available.
5	Access denied.
6	Handle invalid or not open.
7	File was too small.
8	Record number out of range.
9	Record offset contains non-numeric charaters.
10	Record number contains non-numeric characters.

Table 1

Findoffs.PRG is a utility program that finds the offset of a field within a database record. Though we are dealing only with memo fields in this article, Findoffs can be used to find the offset of any field in a database. It is called with the syntax

```
offset = 0
DO Findoffs WITH "<filename>",;
                 "<fieldname>", offset
```

The first two parameters may be character-type memory variables or literals holding the name of the file and the field, respectively. The last parameter must be an initialized numeric memory variable, since Findoffs puts the result there.

Findoffs returns the offset to the specified field in the numeric memory variable. If the variable is negative, however, an error has occurred. The two possible error codes are shown in Table 2.

Example Program: Memotest

Memotest.PRG is an example program that uses both Ismemo.BIN and Findoffs.PRG. It asks for the name of a database and the name of a memofield within that database. It then walks through the database and reports whether a memo exists for each record.

Memotest is run from the dot prompt with the command

```
DO Memotest
```

Setup

To use Ismemo, you must first enter the source code into a text file named Ismemo.ASM. The code for this program is very similar to the code found in Mem2char.ASM, in the August 1987 ***TechNotes***. If you have the source code for that program, you could make a copy of it and edit from there.

Enter the lines for Assemble.BAT in a separate file. Assemble.BAT is a generic batch file that you can use to assemble any .BIN file.

To assemble Ismemo after the files Ismemo.ASM and Assemble.BAT have been created, type

```
ASSEMBLE Ismemo
```

at the DOS prompt. Masm.EXE, Link.EXE, and Exe2bin.EXE all must be available in the current directory or in your DOS path. You should type only the .ASM filename, not the extension.

The source code for Findoffs.PRG and Memotest.PRG should be entered into two separate text files.

Error Number	Meaning
-1	Database file not found.
-2	Field not found within database.

Table 2

ISMEMO.ASM

```
; ISMEMO.ASM
; Program..: Ismemo.ASM
; Author...: Kenneth N. Getz
; Date.....: January 1, 1988
; Notes....: Ismemo.ASM tests for the existence of a memo field for a
;            specific record in a specific file.  The routine is
;            called with the following parameter string stored in a
;            memory variable:
;
;               STR(offset of the memo field, 4) + ;
;               STR(record number, 10) +;
;               "database filename"
;
;            If no error occurs, the first character in the memory
;            variable parameter will be a space (CHR(32)) and the
;            second character will be either a "Y" or an "N", depending
;            on whether a memo field was found.
;
;            If an error occurs, one of the following codes will be
;            returned in the first byte of the memory variable.  The
;            first 5 (2 - 6) are assigned by DOS, and the last four
;            are assigned by Ismemo.
;
;            Errors:
;               CHR(2) : File not found.
;               CHR(3) : Path not found or file doesn't exist.
;               CHR(4) : No file handle available.
;               CHR(5) : Access denied.
;               CHR(6) : Handle invalid or not open.
;               CHR(7) : The file was too small.
;               CHR(8) : The record number was out of range.
;               CHR(9) : The record offset passed in was non-numeric.
;               CHR(10): The record number passed in was non-numeric.
;               Any other character : Unspecified file error.
;
;          You will find that the code for this program is very similar
;          to the code found in Mem2char.ASM from the August 1987
;          issue of TechNotes.  If you've already entered that program,
;          you may want to make a copy of it and start editing from there.
;
;*****************************************************************************
DOS     EQU     21h             ; Call to DOS.
CREATE  EQU     3Ch             ; Create a file.
OPEN    EQU     3Dh             ; Open a file.
CLOSE   EQU     3Eh             ; Close a file.
READ    EQU     3Fh             ; Read from a file.
WRITE   EQU     40h             ; Write to a file.
MOVE    EQU     42h             ; Move file pointer.
X_SIZE  EQU     7               ; Indicates the file was too small.
X_RNUM  EQU     8               ; Indicates record number out of range.
X_FLD   EQU     9               ; Record offset non-numeric.
X_REC   EQU     10              ; Record number non-numeric.
;-----------------------------------------------------------------------------
SAVE   MACRO  R1,R2,R3,R4,R5,R6,R7,R8
;
; This macro takes a parameter list and pushes each parameter register, in the
; order given, onto the stack.
;
;      USAGE:  SAVE AX,BX,CX...
```

```
          IRP     X,<R1,R2,R3,R4,R5,R6,R7,R8>
                  IFNB    <X>
                          PUSH X
                  ENDIF
          ENDM
          ENDM
;-------------------------------------------------------------------------------
RESTORE MACRO  R1,R2,R3,R4,R5,R6,R7,R8
;
; This macro takes a parameter list and pops each parameter register, in the
; opposite of the order given, from the stack.
;
;      USAGE:  RESTORE AX,BX,CX...
;
          IRP     X,<R8,R7,R6,R5,R4,R3,R2,R1>
                  IFNB    <X>
                          POP X
                  ENDIF
          ENDM
          ENDM
;-------------------------------------------------------------------------------
GETFILE MACRO  MODE
LOCAL  L1,L2,L3
;
; This macro opens a file, putting the file handle into the variable HANDLE.
; If an error occurs, the error code is sent back to dBASE in the first
; character of the passed parameter.  The ASCIIZ string for the file
; name is passed in by dBASE in the DS:DX pair.
;
;      USAGE:  GETFILE CREATE
;      USAGE:  GETFILE READ
;
          XOR     AX,AX                   ; Clear out AX.
          MOV     DX,OFFSET CS:FILENM     ; INT 21h needs address in DS:DX
          MOV     AH,MODE                 ; Get the correct mode in AH.
          MOV     AL,2
          CLC
          INT     DOS                     ; Call DOS.
          JC      L2
                                          ; Check for carry flag set (ERROR!)
L1:       MOV     CS:HANDLE,AX            ; If OK, move handle to storage.
          JMP     L3
L2:       MOV     CS:ERROR,AX             ; Otherwise, store error code.
L3:       NOP
          ENDM
;-------------------------------------------------------------------------------
DOSFILE MACRO  @AH,@AL,@BX,@CX,@DX
;
; This macro simplifies calling DOS for all interrupt functions.
;
;     USAGE:  DOSFILE READ,0,HANDLE,1,<OFFSET CHECK>
;
          MOV     AH,@AH
          MOV     AL,@AL
          MOV     BX,@BX
          MOV     CX,@CX
          MOV     DX,@DX
          INT     DOS
          ENDM
;*******************************************************************************
```

```
CODE    SEGMENT BYTE PUBLIC    'CODE'
;-------------------------------------------------------------------------
ISMEMO    PROC    FAR
          ASSUME  CS:code

START:    JMP     ENTRY
;=========================================================================
          DB      13,'Ismemo.BIN',10,13
          DB      'Written by Kenneth N. Getz.',10,13
          DB      '(C) Copyright 1988, Ashton-Tate',10,13
          DB      26

TEN       DW      10                    ; Constant used for multiplication.
CHECK     DB      10 DUP(0)             ; Data area for memo field check.
RECOFFS   DW      0                     ; Offset in record for memo field.
RECPOS    DB      4  DUP(0)             ; ASCII offset for memo field.
RECNO     DB      10 DUP(0)             ; ASCII record number.
FILENM    DB      65 DUP(0)             ; Filename.
HANDLE    DW      0                     ; File handle.
ERROR     DW      0                     ; Error flag.
POS1      DW      ?                     ; Low-word of the file offset.
POS2      DW      ?                     ; High-word of the file offset.
HEADER    STRUC                         ; Structure for header information.
          VERSION   DB      0
          DATE      DB      3 DUP(0)
          NUMRECS   DD      0
          HEADSZ    DW      0
          RECLEN    DW      0
          RESRVED   DB      20 DUP(0)
HEADER    ENDS
HEAD      HEADER  <>                    ; Initialization of structure.
ISMEMO    DB      ' '                   ; Data storage - is there a memo field?
;=========================================================================
ENTRY:    SAVE    AX,BX,CX,DX,SI,DI     ; Save the environment.
          PUSH    DS                    ; Save dBASE's Data Segment.
          MOV     HANDLE,0              ; Initialize file handle to 0.
          MOV     ISMEMO,32             ; Initialize the memo flag.
          CALL    PARAMS                ; Get the passed parameters.
          PUSH    CS                    ; Make DS = CS.
          POP     DS
          MOV     ERROR,32              ; Initialize error flag to be a space.
          GETFILE OPEN                  ; Open the necessary file.
          CMP     ERROR,32              ; See if an error occurred.
          JNE     OUT                   ; If so, quit.
          CALL    GETHD                 ; Read in the file header.
          CMP     ERROR,32              ; See if an error occurred.
          JNE     OUT                   ; If so, quit.
          CALL    MAKECHG               ; Do the actual work.
OUT:      POP     DS                    ; Restore dBASE's DS register.
          CMP     CS:HANDLE,0           ; See if we ever opened a file.
          JZ      NO_CLS                ; If not, don't close one.
          CALL    CLSFILE               ; Otherwise, close the file.
NO_CLS:   MOV     AL,BYTE PTR CS:ERROR  ; Send back the error byte.
          MOV     [BX],AL               ; Tell dBASE about error.
          MOV     AL,BYTE PTR CS:ISMEMO ; Was a memo field found?
          MOV     [BX+1],AL             ; Tell dBASE about it.
          RESTORE AX,BX,CX,DX,SI,DI     ; Restore the environment.
          RET                           ; And return to dBASE.
  ISMEMO  ENDP
;-------------------------------------------------------------------------
PARAMS  PROC    NEAR
```

```
;
; This procedure fills the data area with arguments passed on the command
; line from dBASE.  This routine is very dependent on the order of the
; variables in the data area.
;
          SAVE    AX,CX,DX,DS             ; Save the environment.
          MOV     CL,BYTE PTR [BX-1]      ; Get dBASE's param string length.
          XOR     CH,CH                   ; Clear the high byte.
          DEC     CX                      ; Don't want the last NULL byte.
          MOV     SI,BX                   ; Point SI to the string.
          MOV     DI,OFFSET RECPOS        ; Point DI to the buffer area.
          REP     MOVSB                   ; Copy it all in to the code area.
          XOR     AL,AL                   ; Zero out AL
          STOSB                           ; Make sure last byte is 0.
          RESTORE AX,CX,DX,DS             ; Restore the environment.
          RET
PARAMS    ENDP
;---------------------------------------------------------------------------
GETHD  PROC   NEAR
;
; This procedure reads the file header into the structure HEAD.
; If an error occurs (32 bytes couldn't be read), ERROR will contain
; X_SIZE.
;
          SAVE    AX,BX,CX,DX             ; Read 32-byte file header.
          DOSFILE READ,0,HANDLE,32,<OFFSET HEAD>
          CMP     AX,CX                   ; See if we read all 32 bytes.
          JE      HD_OK                   ; If so, we're OK.
          MOV     ERROR,X_SIZE            ; Otherwise, we have an error.
HD_OK:    RESTORE AX,BX,CX,DX             ; Restore all registers.
          RET
GETHD     ENDP
;---------------------------------------------------------------------------
MAKECHG PROC   NEAR
;
; This procedure actually performs the work.  It reads the necessary byte
;
          SAVE    AX,BX,CX,DX
          MOV     ISMEMO,32               ; Initialize it to be a space.
          MOV     SI,OFFSET RECNO         ; Point SI to the ASCII record number.
          CALL    CVRTDW                  ; Put record number in DX:AX.
          CMP     ERROR,32                ; Check for error.
          JNE     NO_CHG                  ; If there is one, quit.
          CMP     DX,WORD PTR HEAD.NUMRECS+2  ; DX better be less.
          JBE     HI_OK                   ; If DX OK, go on.
          MOV     ERROR,X_RNUM            ; Otherwise, set error.
          JMP     NO_CHG                  ; And quit.
  HI_OK:  CMP     AX,WORD PTR HEAD.NUMRECS    ; AX better be less.
          JBE     LO_OK                   ; IF AX OK, go on.
          MOV     ERROR,X_RNUM            ; Otherwise, set error.
          JMP     NO_CHG                  ; And quit.
  LO_OK:  DEC     AX                      ; Decrement to get correct offset.
          MOV     DI,HEAD.RECLEN          ; Get the record length.
          CALL    MULT                    ; Multiply to get file offset.
          ADD     AX,HEAD.HEADSZ          ; Add on header size.
          ADC     DX,0                    ; Update high word if necessary.
          CALL    CVRT                    ; Puts memo offset into BX.
          CMP     ERROR,32                ; See if error occurred.
          JNE     NO_CHG                  ; If so, quit.
          ADD     AX,BX                   ; Add memo offset to file offset.
          ADC     DX,0                    ; Update high word if necessary.
```

```
          INC     AX                      ; Skip over first byte (' ').
          ADC     DX,0                    ; Update high word if necessary.
  SIZE_OK:
          MOV     POS2,DX
          MOV     POS1,AX                 ; Store away the file position.
          DOSFILE MOVE,0,HANDLE,POS2,POS1         ; Go to that file offset.
          DOSFILE READ,0,HANDLE,10,<OFFSET CHECK>  ; Read the bytes.
          CALL    MEMTEST                 ; Check for a memo field.
  NO_CHG:
          RESTORE AX,BX,CX,DX             ; Restore all registers.
          RET
MAKECHG   ENDP
;-------------------------------------------------------------------------
CLSFILE   PROC    NEAR
;
; This procedure closes the file referred to by HANDLE.  If an error
; occurred while closing the file, the error code is passed back to
; dBASE in the first character of the calling parameter.
;
          SAVE    AX,BX
          MOV     BX,CS:HANDLE            ; Get the file handle.
          MOV     AH,CLOSE                ; Set up for Close.
          CLC                             ; Clear the Carry Flag.
          INT     DOS                     ; Call DOS.
          JNC     CLS_OK                  ; If Carry Flag still clear, we're OK.
          MOV     CS:ERROR,AX             ; Otherwise, send error back.
CLS_OK:
          RESTORE AX,BX
          RET
CLSFILE ENDP
;-------------------------------------------------------------------------
CVRT    PROC    NEAR
;
; Ends up with string pointed to by SI converted to binary in BX.
; This routine initializes BX to zero, and stops when it gets to any
; non-numeric character.  This means that if you pass in a string like
; "QRST", this procedure will return a 0 in BX.  If you pass in "123q",
; it will place 123 in BX.  It does skip over leading spaces.
;
          SAVE    AX,CX
          SUB     AX,AX                   ; Clear out AX.
          SUB     BX,BX                   ; Clear out BX.
          MOV     SI,OFFSET RECPOS        ; Point SI to the parameter storage.
          MOV     CX,4                    ; Set loop counter.
NUMLP1:   CMP     BYTE PTR [SI],32        ; See if we got a space.
          JE      SPACE1                  ; If so, go on to next character.
          CMP     BYTE PTR [SI],'0'       ; See if character less than '0'.
          JB      ERR1                    ; If so, we have an error.
          CMP     BYTE PTR [SI],'9'       ; See if character greater than '9'.
          JA      ERR1                    ; If so, we have an error.
          MUL     TEN                     ; If no error, multiply AX by 10.
          MOV     BL,[SI]                 ; Get the next input character.
          SUB     BX,'0'                  ; Subtract ASCII offset.
          ADD     AX,BX                   ; Add in the new digit.
SPACE1:   INC     SI                      ; Go on to the next character.
          LOOP    NUMLP1                  ; Loop back for more.
          MOV     BX,AX                   ; Store the answer in BX.
          JMP     NUMDN1
ERR1:     MOV     ERROR,X_FLD             ; Set up error.
NUMDN1:   RESTORE AX,CX
          RET
```

```
CVRT     ENDP
;-------------------------------------------------------------------------
CVRTDW   PROC   NEAR
;
; This procedure converts the 10-digit ASCII string pointed to by SI into
; a double-word integer in DX:AX.
;
; The same rules apply here as in CVRT.  If you pass a non-numeric string,
; CVRTDW will return 0 (this is not normally a legal record number).
; Essentially, this procedure halts and returns what it currently has
; calculated as soon as it finds either the end of the string or a non-
; numeric character.
;
          SAVE    BX,CX,DI
          SUB     AX,AX                   ; Clear out AX.
          SUB     DX,DX                   ; Clear out DX.
          MOV     CX,10                   ; Set loop counter for 10 digits.
NUMLP2:   CMP     BYTE PTR [SI],32        ; See if we got a space.
          JE      SPACE2                  ; If so, go on to next character.
          CMP     BYTE PTR [SI],'0'       ; See if digit less than '0'.
          JB      ERR2                    ; If so, quit.
          CMP     BYTE PTR [SI],'9'       ; See if digit > '9'
          JA      ERR2                    ; If so, quit.
          MOV     DI,TEN                  ; Get set for multiply.
          CALL    MULT                    ; Multiply DX:AX by DI.
          XOR     BH,BH                   ; Clear out BH.
          MOV     BL,[SI]                 ; Get the next digit.
          SUB     BX,'0'                  ; Subtract ASCII offset.
          ADD     AX,BX                   ; Add in the new digit.
          ADC     DX,0                    ; Update DX if necessary.
SPACE2:   INC     SI                      ; Get to next character.
          LOOP    NUMLP2                  ; Go back for more.
          JMP     NUMDN2                  ; No error.
ERR2:     MOV     ERROR,X_REC             ; Send back error.
NUMDN2:   RESTORE BX,CX,DI
          RET
CVRTDW    ENDP
;-------------------------------------------------------------------------
MULT      PROC    NEAR
;
; Multiply the number in DX:AX by the word in DI and put in DX:AX.
;
          PUSH    DX                      ; Save current DX.
          MUL     DI                      ; Multiply AX by DI.
          MOV     BX,DX                   ; Move the carry portion into BX.
          POP     DX                      ; Restore the old DX.
          PUSH    AX                      ; Save the current AX.
          MOV     AX,DX                   ; Get the old DX ready to multiply.
          MUL     DI                      ; Multiply by DI.
          ADD     AX,BX                   ; Add on the carry from the first MUL.
          MOV     DX,AX                   ; Move the new high word into DX.
          POP     AX                      ; Get the current low word back.
          RET
MULT      ENDP
;-------------------------------------------------------------------------
MEMTEST   PROC   NEAR
;
; This procedure looks through the 10-byte memo-field information string
; looking for something besides spaces.  If it finds anything, it has
; to assume that there is a memo field attached and sends this information
; back.  If all spaces are found, there is no memo field.
```

```
          SAVE    AX,CX,SI
          MOV     SI,OFFSET CHECK         ; Point SI to memo information.
          MOV     ISMEMO,'N'              ; Initialize to No.
          CALL    CVRTDW                  ; Convert memo pointer to double int.
          CMP     DX,0                    ; See if high word is 0.
          JNE     MDONE                   ; If not, we have a memo field.
          CMP     AX,0                    ; See if low word is 0.
          JE      ISDONE                  ; If it is, we don't have a memo field.
  MDONE:  MOV     ISMEMO,'Y'              ; Change ISMEMO to 'Y'.
  ISDONE: RESTORE AX,CX,SI
          RET
  MEMTEST ENDP
;-------------------------------------------------------------------------------
  CODE    ENDS
          END     START
```

ASSEMBLE.BAT

```
REM Assemble.BAT
ECHO OFF
CLS
IF "X%1" == "X" GOTO ERROR3
MASM %1,%1,%1.LST;
IF ERRORLEVEL = 1 GOTO ERROR1
LINK %1,%1,NUL;
IF ERRORLEVEL = 1 GOTO ERROR2
ECHO It is proper to ignore the previous error message.
EXE2BIN %1.EXE %1.BIN
GOTO EXIT
:ERROR1
ECHO ASSEMBLER ERROR!
GOTO EXIT
:ERROR2
ERASE %1.EXE
ECHO LINKER ERROR!
GOTO EXIT
:ERROR3
ECHO Please try again -- ASSEMBLE requires a filename.
ECHO Example: ASSEMBLE FOO
:EXIT
```

FINDOFFS

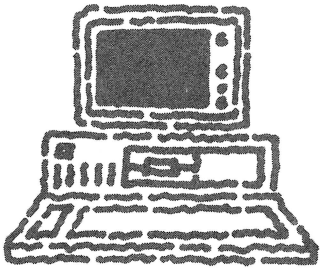
```
* Program...:  Findoffs.PRG
* Author....:  Kenneth N. Getz
* Date......:  January 1, 1988
* Notes.....:  Findoffs.PRG finds the offset within a record for the given
*              field.  You call Findoffs with the command line:
*
*                 offset = 0
*                 DO Findoffs WITH "<filename>", "<fieldname>", offset
*
*              The memory variable "offset" will hold the correct numeric
*              value upon return.
*
*              If an error occurs, Findoffs will return one of the following
*              error codes in the variable offset:
*
*                 -1 : Database file not found.
```

```
*                -2 : Fieldname not found within database file.
*
PARAMETERS filename, fieldname, offset
*
*
* ---Initialize the necessary variables.
filename  = TRIM(filename)
fieldname = UPPER(TRIM(fieldname))
offset    = 0
*
* ---If the file exists, use it, otherwise return an error.
IF FILE("&filename..DBF")
  USE &filename
ELSE
   offset = -1
   RETURN
ENDIF
*
* ---Build a structure extended file and USE it.
COPY STRUCTURE EXTENDED TO &filename..EXT
USE &filename..EXT
*
* ---Add up the field lengths ahead of the field in question.
DO WHILE Field_name <> fieldname .AND. .NOT. EOF()
  offset = offset + Field_len
  SKIP
ENDDO
USE
ERASE &filename..EXT
*
* ---If the field is not found, return an error.
IF EOF()
  offset = -2
ENDIF
* EOP Findoffs.PRG
```

MEMOTEST

```
* Program..:  Memotest.PRG
* Author...:  Kenneth N. Getz
* Date.....:  January 1, 1988
* Notes....:  Memotest.PRG is a test program for Ismemo.BIN.  It walks through
*             a database file and reports on the memo field status of each
*             record.
*
*             Requires the program Findoffs.PRG to calculate the offset
*             within the record of the requested memo field.
*
SET TALK OFF
LOAD Ismemo
CLEAR
*
* ---Initialize variables.
paramstr  = SPACE(75)
filename  = SPACE(64)
fieldname = space(10)
offset    = 0
*
* ---Get the file and field names.
@ 1,1 SAY "Filename   : " GET filename
@ 2,1 SAY "Memo field : " GET fieldname
```

```
READ
CLEAR
*
* ---Figure the offset within the record.
DO Findoffs WITH filename, fieldname, offset
IF offset < 0
  ? UPPER(TRIM(IIF(offset = -1, filename, fieldname))) +;
                  " not found.  Halting..."
   RETURN
ENDIF
USE &filename
DO WHILE .NOT. EOF()
   *
   * ---Build the parameter string.
   paramstr = STR(offset, 4) + STR(RECNO(), 10) + DBF()
   CALL Ismemo WITH paramstr
   *
   * ---A space indicates no error.  Anything else means something went
   * ---wrong.
   IF LEFT(paramstr, 1) <> CHR(32)
      ? "Error number " + STR(ASC(LEFT(paramstr, 1)), 2, 0) +;
         " at record " + LTRIM(STR(RECNO(), 10)) + "."
      EXIT
   ELSE
      *
      * ---If the second character is a "Y", there was a memo field.
      ok = (SUBSTR(paramstr, 2, 1) = "Y")
    ? "RECORD " + LTRIM(STR(RECNO(), 10)) + " DOES "
    ?? IIF(ok, SPACE(4), "NOT ") + "CONTAIN A MEMO."
     SKIP
  ENDIF
ENDDO
* EOP Memotest.PRG
```



A Soft Record-Locking Subroutine for the dBASE III PLUS LAN Pack

By Richard Bunter and Chuck Litzell

The Problem

The dBASE III PLUS Administrator permits multiple workstations to access the same database file simultaneously. A necessary feature of such a multiuser system is record locking. Without record locking, users on different workstations might attempt to change the same record at the same time, which would result in a network collision and corrupted data. When a record is locked, it can be accessed only by the workstation that placed the lock.

The record-locking mechanism provides an essential safeguard against such collisions. In the process, though, it can impede the efficiency of the other workstations on the network. Clearly, you don't want multiple workstations changing a record simultaneously, but why shouldn't other users at least be able to look at a record that is already being edited by somebody else? In other words, should locked records be available read-only to other users?

Customer Service, Inc, has a network with 15 workstations. The dBASE III PLUS Administrator is installed on the server, and all the workstations access it regularly. The most active database file on the server is the Customer file. Almost all programs in the system must have access to the information in the Customer file. All maintenance to the Customer file – additions, changes or deletions – is accomplished through a single edit program.

Joe is using the Customer maintenance program to edit a customer's record. The program has locked the record because Joe is in the middle of changing some fields. His phone rings. It's a customer inquiry that sends Joe off hunting for a missing purchase order.

In another part of the office, Melinda is about to run an updated Customer listing. As soon as she tries to start the report, the system prompts: "Record is in use by another."

Joe finishes his phone call with the customer and decides it's a good time to break for lunch.

Melinda attempts to run the report every few minutes without success.

Melinda knows there is a possibility that someone has inadvertently left a computer with a locked record, but there is an equal probability that, at any given moment, one or more of the fifteen workstations are actively editing records.

So we have a dilemma. Users cannot change records that are not locked, and locked records prevent other users from getting their work done. Is there a solution?

One way to ease the situation is to make sure records are locked only momentarily. A record *must* be locked when the information in it is changed; there is no way around this. But if changes are entered into memory variables, the record can remain unlocked until the data entry is complete. Then the record is locked, updated with REPLACE statements, and unlocked.

The potential flaw in this game plan is that if the record is not locked, two or more users can be updating the same record unknowingly. Only changes made by the last one to finish will be recorded permanently.

The Solution: Softlock

Soft locking records makes the strategy workable. When a user soft locks a record he or she is telling other users, "You can look, but don't touch." Since the record is not physically locked, anyone can access it; a user cannot bring the system to a halt because she was distracted in the middle of a record edit.

Soft locking is not a function of the dBASE III PLUS Administrator. It is implemented through a program that processes record-locking requests. Soft locking requires that you add a logical field, Inuse, to the database structure. When a program needs "update access" to a record, the Softlock subroutine is called.

Softlock places a status code in the logical memory variable lockstat, which must be declared by the calling program. If lockstat is .T. when Softlock returns, the program can proceed to load fields into memory variables for editing. When the edit is complete the program can lock the record, REPLACE the updated fields, REPLACE Inuse with .F., and then unlock the record.

In pseudocode, Softlock follows these steps:

```
Attempt to lock the record.
If record was locked and Inuse field is not set to true,
   set Inuse field to true, and
   set lock status code to true;
otherwise
   set lock status code to false.
Unlock the record.
```

Calling Softlock from an Edit Program

A program that edits database records should call Softlock after positioning to the record to be edited. The database structure must include a logical field called Inuse. Three memory variables must also be established before the call:

msgline a numeric variable that contains the row number on which Softlock will display messages

maxtries number of times to attempt RLOCK() before allowing user to abort

lockstat a logical variable that contains the status of the record lock when Softlock returns to the calling program. If lockstat is set to .T., the record's fields can be moved to memory variables for editing. If lockstat is .F., the record could not be locked, or is marked Inuse by another user.

Testedit.PRG

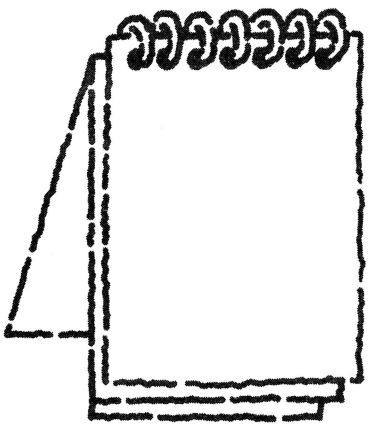
```
* Program ...: Testedit.PRG
* Author ....: Chuck Litzell
* Date ......: January 1, 1988
* Versions ..: dBASE III PLUS Administrator, 1.0 and 1.1
* Notes .....: Sample program to demonstrate use of Softlock.PRG.
*
*           Testedit edits Clients.DBF, found on the dBASE III
*           PLUS Sample Programs and Utilities diskette.
*           No indexes are used for this test. If your program
*           will edit a database file with indexes, refer to
*           "LAN Index Integrity," in the November 1987TechNotes.
*
SET TALK OFF
SET STATUS ON
SET EXCLUSIVE OFF
* ---Memory variables required for Softlock.
msgline  = 21
maxtries = 5
lockstat = .F.

 USE Clients

 DO WHILE .T.
     recnum = 0
    @ msgline, 28 SAY "Record number to edit" ;
                  GET recnum ;
                  PICTURE "99999" ;
                  RANGE 0, RECCOUNT()
     READ
     @ msgline, 0
     IF recnum = 0
        EXIT
     ENDIF
     GO recnum
     DO Softlock
     IF .NOT. lockstat
        @ msgline, 0
        @ msgline, 18 SAY "Record is in use by another.Press any key."
         DO WHILE INKEY() = 0
```

```
            ENDDO
            @ msgline, 0
         ELSE
            mfirst   = Firstname
            mlast    = Lastname
            maddress = Address
            mcity    = City
            mstate   = State
            mzipcode = Zipcode
            mphone   = Phone
            CLEAR
            @  2, 10 SAY "Record ......." + LTRIM(STR(RECNO()))
            @  4, 10 SAY "First Name ..." GET mfirst
            @  5, 10 SAY "Last Name ...." GET mlast
            @  6, 10 SAY "Address ......" GET maddress
            @  7, 10 SAY "City ........." GET mcity
            @  8, 10 SAY "Zip Code ....." GET mzipcode
            @  9, 10 SAY "Phone ........" GET mphone
            READ
            @ msgline,  28 SAY "Please wait ... Updating"
            locktries = 0
            * ---Note:  The following code is similar to
            * ---the code used to lock a record in Softlock.PRG.
            * ---In practice, you could move this routine to a separate
            * ---procedure called by your edit program and Softlock.PRG.
            DO WHILE .NOT. RLOCK()
               locktries = locktries + 1
               IF locktries > maxtries
                  ans = .T.
                  @ msgline, 0
                  @ msgline, 22 SAY "Record is in use by another.Retry?";
                                GET ans;
                                PICTURE "Y"
                  READ
                  @ msgline, 0
                  IF ans
                    @ msgline, 28 SAY "Please wait ... Updating"
                    locktries = 0
                    LOOP
                  ELSE
                    @ msgline, 22 SAY "Record not updated.Press any key."
                    DO WHILE INKEY() = 0
                    ENDDO
                    @ msgline, 0
                    EXIT
                  ENDIF
               ENDIF
            ENDDO

      * ---   Record is locked.
            REPLACE Firstname WITH mfirst, Lastname WITH mlast,;
                    Address WITH maddress, City WITH mcity,;
                    State WITH mstate, Zipcode WITH mzipcode,;
                    Phone WITH mphone, Inuse WITH .F.
            UNLOCK
         ENDIF
      ENDDO
      CLOSE DATABASES
      * EOP Testedit.PRG
```

Softlock.PRG

```
* Program ...: Softlock.PRG
* Author .....: Richard Bunter
* Date .......: January 1, 1988
* Versions ..: dBASE III PLUS Administrator, 1.0 and 1.1
* Note(s) ...: Attempts to lock the current database record.
*
*              If record can be locked and hasn't been claimed by
*              another user (.NOT. Inuse), the Inuse field
*              is replaced with .T.
*
*              Calling program must have established three variables:
*                 msgline  :=  Row number for message displays
*                 maxtries :=  Number of times to attempt record
*                              lock before asking user
*                 lockstat :=  Logical variable to hold lock status
*
*              Return:
*                 lockstat is .T.:  Record is available for edit, and
*                                   Inuse is set to .T.
*                 lockstat is .F.:  Record is not available for edit.
*
*                 Record is UNLOCKed on exit.
*
message   = "Record is in use by another ... Retry?"
locktries = 0
lockstat  = .T.
*
* ---Attempt to lock the record.
*
DO WHILE .NOT. RLOCK()
   locktries = locktries + 1
   *
   * ---If we haven't been able to lock the record after maxtries
   * ---attempts, ask the user if he wants to keep trying.
   IF locktries >= maxtries
        ans = .T.
        @ msgline, 0
        @ msgline, (80 - LEN(message) + 2) / 2) GET ans PICTURE "Y"
```

```
            READ
            @ msgline, 0
            IF ans
               *
               * ---Set counter to 0 and try again.
               *
               locktries = 0
               LOOP
            ELSE
               *
               * ---Can't lock record; quit trying.
               *
                lockstat = .F.
                EXIT
            ENDIF
         ENDIF
ENDDO
*
* ---lockstat is still .T. if we were able to lock the record.
* ---Now, see if the record is Inuse.
*
IF lockstat .AND. .NOT. Inuse
   *
   * ---Record is available.
   *
   REPLACE Inuse with .T.
ELSE
   * ---Someone else is using the record.
   *
   lockstat = .F.
ENDIF


* ---Leave record unlocked while we work on it.
UNLOCK
RETURN
* EOP: Softlock.PRG
```



SetHelpOn

Printing with Special Effects

By Naoma Stow

Although computers are supposed to reduce the amount of paper that modern offices generate, sooner or later someone is going to want a nicely printed report based on the information in your database file. How, then, can you get dBASE III PLUS to use your printer's special formatting features, such as underlining, italic, and boldface?

Unlike MultiMate or Framework II, dBASE III PLUS doesn't use printer drivers, which are special files that tell software how to send information to specific printer models. As a consequence, to get special styles of printing, you need to know how to get dBASE III PLUS to tell the printer what to do. The way to do this is to send the appropriate control codes to your printer.

Different printer manufacturers use different control codes to signify special features. How can you find out what the codes are for your printer? If you look in the manual that came with the printer, there should be a table that lists the various styles and the codes required to produce them.

Working with Limited Text

Let's start with a simple task: applying a special printing style to a specific piece of text, such as a title or a heading. To *underline* a title, for instance, you would start by finding out what the codes for *underlining* with your printer are. While this information should be in your printer manual, there's no guarantee that it will be comprehensible. A typical printer manual might contain information on italics like that in Table 1.

This is not so cryptic as it might first appear. Two codes are needed: one to tell the printer to select italics (Esc 4) and another to tell it to stop (Esc 5). The numbers listed under the heading "decimal" are the ASCII codes (in decimal format) for the escape character and the 4 or 5 character. These ASCII codes are a near-universal standard. Thus, any printer would interpret the sequence of decimal 27 and decimal 52 as Esc 4. The real question is what the printer interprets Esc 4 to mean. Some printers, such as an Epson

command	decimal	hexadecimal	function
ESC 4	27 52	1B 34	select italic mode
ESC 5	27 53	1B 35	cancel italic mode

Table 1

FX-80, would interpret it to mean "start italics." Other printers might attach a completely different meaning or no meaning.

Thus, to italicize a specific phrase or word on an Epson FX-80, you first send an escape character to the printer and then immediately follow by sending the character "4." Then you send the text that you want to print in italics. Finally, to turn the italics off, you send another escape character, followed this time by the character "5."

How do you "send" characters from within dBASE III PLUS? Well, characters like "4" and "5" are easy; you print them just like any other text. Nonprintable characters, such as escape, are only a little trickier. The dBASE III PLUS function CHR() is used for these. The CHR() function lets you refer to any character, printable or otherwise, by its decimal ASCII code. Thus if you know that the ASCII code (in decimal) for escape is 27 (which you do – the printer manual told you so), then sending CHR(27) to the printer is the same as sending it an escape character.

Here's exactly how you would do it:

```
SET PRINT ON
? CHR(27)+ "4" + "This is in italics" + ;
  CHR(27)+ "5"
```

The result should be

This is in italics

Note: If you don't know what the ASCII equivalent for a character is, you can look it up in an ASCII chart. There's one in the Appendices of the *Learning and Using dBASE III PLUS* manual.

General Print Style

Suppose you want to print your entire report in a special print style, such as compressed print. Some printers will let you set different modes of printing by changing a DIP switch or, even more conveniently, by pressing one of the printer's control buttons. If your printer doesn't allow you to set compressed mode in those ways, or if you'd simply rather leave it up to dBASE III PLUS to tell the printer what to do, control codes once again can come to the rescue.

For many printers, the control code for starting compressed print is the nonprintable character represented by ASCII decimal code 15. You can check your printer manual to determine whether this is the case for your printer. Assuming that decimal 15 is the code, this is how you could tell your printer, from the dBASE III PLUS dot prompt, to switch to compressed print:

```
SET PRINT ON
? CHR(15)
SET PRINT OFF
```

After you send these commands, any text that you send to the printer appears in the compressed style. The compressed style then remains in effect until the printer is turned off or until you send the control code for setting compressed off.

Of course, if your printer's code for compressed print is something other than decimal 15, you can substitute that number in the CHR() function and get the same result.

Memory Variables

Control codes are an effective way to control your printer from dBASE III PLUS, but they come up a little short in the convenience and readability departments. Within your dBASE programs, you can make control codes easier to use and a lot more readable by storing them to memory variables. This technique can save both time and typing.

Note: Memory variables are temporary storage locations for information. In this case, we're using them to store the control codes. For a more detailed explanation of how memory variables work, see Chapter 3, "Using Memory Variables," in the *Programming with dBASE III PLUS* manual.

You can store a control code to a memory variable in a couple of ways:

```
STORE CHR(27) + "4" TO pr_it_on
```

or

```
pr_it_off = CHR(27) + "5"
```

What these lines of code are doing is taking the control code values and assigning them to the memory variables pr_it_on and pr_it_off, which can now be used to turn *underlining* on and off, respectively. With these memory variables defined in this way, typing

```
? pr_it_on + "This is in italics" + pr_it_off
```

will give you the same result as

```
? CHR(27) + "4" + "This is in italics" + ;
  CHR(27) + "5"
```

The difference is that the first line is shorter, easier to type, and definitely easier to understand.

Saving to Memory Files

If you don't want to redefine the control code memory variables every time you load dBASE III PLUS, you can save time by saving groups of memory variables to memory files. Carrying the above example forward a step:

```
SAVE ALL LIKE pr* TO C:Prfile
```

This command causes all the currently defined memory variables starting with "pr" to be saved to a file named Prfile.MEM. You could later use Prfile to restore all the memory variables that were stored in it. The command to do that is

```
RESTORE FROM Prfile
```

You can take this search for convenience one step further by using the memory files that you create to make a complete dBASE III PLUS printer driver system, where a controlling dBASE program accesses various memory files that contain the printer control codes for various printers. To find out how to set up such a system, see "Creating dBASE III PLUS Printer Drivers" by Kent Irwin in the May 1987 ***TechNotes***.

REPORT FORM

What if you want to send data to the printer using dBASE III PLUS's report generator? If you want the entire report to print compressed, you would send the appropriate control codes to the printer as explained earlier.

If, however, you want to use special printing effects in limited report sections, a different approach is called for. An example of this would be in the **Columns:Contents** field. You may add the control codes (or the memory variable containing these codes) for the special effects before the field label and then the control codes to turn the special effect off after the field label. Some styling effects, such as underlining, may require that you use the TRIM() function to avoid the effect being extended for the full field length rather than the data length. Using this technique you can boldface, italicize, or underline your column contents. It should be noted that the control code characters will be counted by dBASE III PLUS when it is determining the position of print. In some cases, you'll be able to compensate for this by using either the SPACE() function or " " to offset some of the position shifts.

The **Page title**, **Columns:Heading**, and **Groups:Group heading** fields will interpret printer control codes literally. This means that you can't directly use special printing effects in dBASE III PLUS report titles or headers. However a workaround is available that will allow you to specify a special printing effect for a title. The syntax for the REPORT command includes a HEADING option that you can use to get the effect you want. If you follow the HEADING option with an expression that contains your control codes and your actual title in the form of memory variables, you can have your title and underline it too. Here's **an example of how this would work with an Epson printer:**

If you define three memory variables like this:

```
mv1 = CHR(27) + "X"
mv2 = CHR(27) + "Y"
mv3 = "This is my Title"
```

and use the following command:

```
REPORT FORM TEST HEADING mv1 + mv3 + mv2 TO PRINT
```

You should get a report heading like this:

```
Page No.    1          This is my Title
01/01/80               Normal Title Position
```

As usual, the control codes count in the heading centering computation (the heading will be two characters closer to the left margin). To get it centered exactly, all you need to do is type the HEADING option like this:

```
HEADING SPACE(2) + mv1 + mv3 + mv2
```

The extra blanks at the front of the expression "push" the title to the correct position.

Further Experimentation

Once you understand how the control codes allow you to command your printer, accessing special effects from dBASE III PLUS is easy. You will not only broaden your palette of available printing formats but also you'll gain a better understanding of how your computer communicates with your printer.

Usage Tips

APPEND FROM ... SDF/ DELIMITED and Dates

Text files in the SDF or DELIMITED formats can be brought into a dBASE III PLUS database file using the commands

```
APPEND FROM <filename> TYPE SDF
```

or

```
APPEND FROM <filename> TYPE DELIMITED
```

If the source file includes dates, you may need to bring the date into a character or numeric field and then perform a conversion process to create a valid dBASE date.

In dBASE III and dBASE III PLUS database files, dates are stored as a string of eight digits:

```
YYYYMMDD
```

For example, January 15, 1988 would be stored as 19880115. If the dates in your text file are in this format, you can bring them directly into a date field. All eight digits must be present for this to work correctly.

If your date is in any other format, you will need to perform at least one conversion step to reformat the date. If the date in the source text file is stored as "MM/DD/YY" or "MM/DD/YYYY", establish a character field with a field length of eight or ten characters, depending on whether the dates have a two- or four-digit year. After you have APPENDed the text file, MODIFY STRUCTURE and change the character field to a date field. dBASE III PLUS will correctly convert the character date to a date field. For dates stored in other formats, follow these general steps:

1. APPEND into a temporary numeric or character field, as appropriate.

2. Add a new date field with MODIFY STRUCTURE.

3. Use the REPLACE command in combination with a dBASE expression that will convert the numeric or character field to a date.

4. Remove the temporary field with MODIFY STRUCTURE.

&

In dBASE III PLUS, a macro function referencing an undefined memory variable is used literally in commands that create files. For example,

```
* ---Release all memory variables.
CLEAR MEMORY
mem = 93459.34
SAVE TO &mvar
```

creates a disk file called "&mvar.MEM" on the default drive. Other dBASE III PLUS commands that create files behave identically. This is true for every version of dBASE III PLUS.

@...GET...PICTURE

When the PICTURE clause of an @...GET...PICTURE command contains conflicting variable length, template, or function arguments, the template may override the function or the value may override the template. Table 1 includes some possibilities.

@...SAY

The original dBASE III PLUS documentation states, in reference to the @...SAY...GET command, "When sending output to the printer, the row and column coordinates may not exceed 255." However, this limitation

applies to dBASE III versions 1.0 and 1.1 only, not to dBASE III PLUS.

COPY DIF

It is undocumented that the .DIF file created by dBASE III PLUS is a version 1 VisiCalc file.

CREATE/MODIFY LABEL

CREATE/MODIFY LABEL does not always produce an error message when invalid label contents are defined. However, LABEL FORM produces the error message "Syntax error in contents expression." when executing the file. To reproduce this, create a label form and enter two valid field names separated by a space. Save and then retrieve the label form. This is true of each version of dBASE III PLUS.

MODIFY STRUCTURE

MODIFY STRUCTURE, on a database file with a .BAK extension, overwrites the original file and causes the new file to contain zero records. When a new file structure is saved, MODIFY STRUCTURE creates a backup of the original database file by renaming the file with a .BAK extension. The new database file is then assigned the extension of the original file. When the backup file has the same name as the new file, it is overwritten and there is no file to APPEND FROM. This occurs with all versions of dBASE III PLUS.

READKEY()

The documentation for READKEY(), states that the key codes listed can be trapped when a full-screen command is exited. However, the keys **Ctrl-Home**, **Ctrl-PgUp**, **Ctrl-PgDn**, and **F1** can be used to exit an active READ command only. They do not exit APPEND, BROWSE, CHANGE, CREATE, EDIT, INSERT, or MODIFY as implied in the documentation.

REPORT FORM

Although not documented anywhere in the manual, when the HEADING option of the REPORT FORM is used, the heading wraps around based on the following formula.

```
(Page width - Margins) - (2 *
length('Page No. xxxxx'))
```

That is, this formula calculates the space left after removing the margins, the space taken by the page number, and matching space on the right side, so that the heading is centered.

RUN/!

If **Ctrl-Break** is pressed while dBASE III PLUS is active, the keystroke is ignored until a RUN/! or QUIT command is executed. When RUN/! is executed, **Ctrl-Break** stops whatever DOS command is called and control returns to dBASE III PLUS. If dBASE III PLUS is started with a batch file and QUIT is executed, the error message "Terminate batch job (Y/N)?" is displayed when the batch file regains control. This occurs in each version of dBASE III PLUS.

SET CATALOG TO

On page U5-156 of the original *Learning and Using dBASE III PLUS* manual, the master catalog created by SET CATALOG TO is incorrectly described as being stored in the root directory of the dBASE III PLUS default drive. The master catalog is stored in the default drive and directory.

Table 1: Some @...GET...PICTURE possibilities

```
mem  = 'AAAA'
mem1 = 100
mem2 = 0
SET DELIMITERS ON
```

mvar	Picture clause	Display	Comment
mem	'xx'	:AA:	Template overrides length.
mem	'xxxxxxxx'	:AAAA:	Length overrides template.
mem	'@S2 xxxx'	:AA:	Scrolls.
mem	'@S3 xxxx'	:AAA:	Scrolls.
mem	'@S5 xxxxxxx'	:AAAA:	Length overrides arguments.
mem	'@S5 x'	:A:	No scroll, template overrides all.
mem1	'@B *******'	:100*	:Function puts value over template and template incorrectly puts spaces where the value would have been placed if there were no function.
mem2	'@Z *******'	:*******:	Template overrides function.

SET CATALOG TO

When a catalog is created or opened with SET CATALOG and SET VIEW is executed, any subsequent attempt to CREATE or MODIFY a LABEL, QUERY, REPORT, SCREEN or FORMAT file returns the erroneous error message "File not CATALOGed since SET CATALOG was OFF when the active database was USEd." After the error message, DISPLAY STATUS reveals that the database file is open and SET CATALOG is ON.

The documentation states that if you have an active catalog and then open a view file, the database files USEd in the view file are not added to the catalog. Furthermore, the documentation also states that catalogs keep track of which auxiliary files belong to which database file. This infers that a view file is a distinct entry in a catalog.

The documentation does not state that INDEX, QUERY, LABEL, REPORT, SCREEN or FORMAT files cannot be added to the catalog with a view file. This is true for all versions of dBASE III PLUS.

STORE

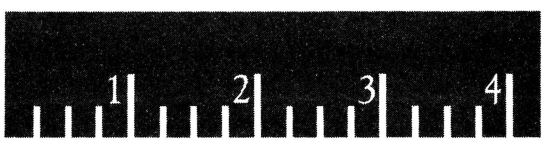
Numeric variables declared with a value of negative billions and no decimal places are stored in scientific notation. This occurs only with negative billions; negative ten billions and negative hundred millions are declared without error, as are positive numbers. For example,

```
* Negative billions are stored in
* scientific notation.
mem = -1234567890
? mem
-.1234E+10
? STR(mem)
-.1234E+10
* Negative hundred millions and
* ten billions are not
* affected.
mem = -123456789
? mem
-123456789
mem = -12345678901
? mem
-12345678901
```

The work-around is to store the number to a variable using the VAL() and STR() functions. For example,

```
SET DECIMALS TO 1
* Must be greater than zero.
mem = VAL(STR(-1234567890,11))
? mem
-1234567890
```

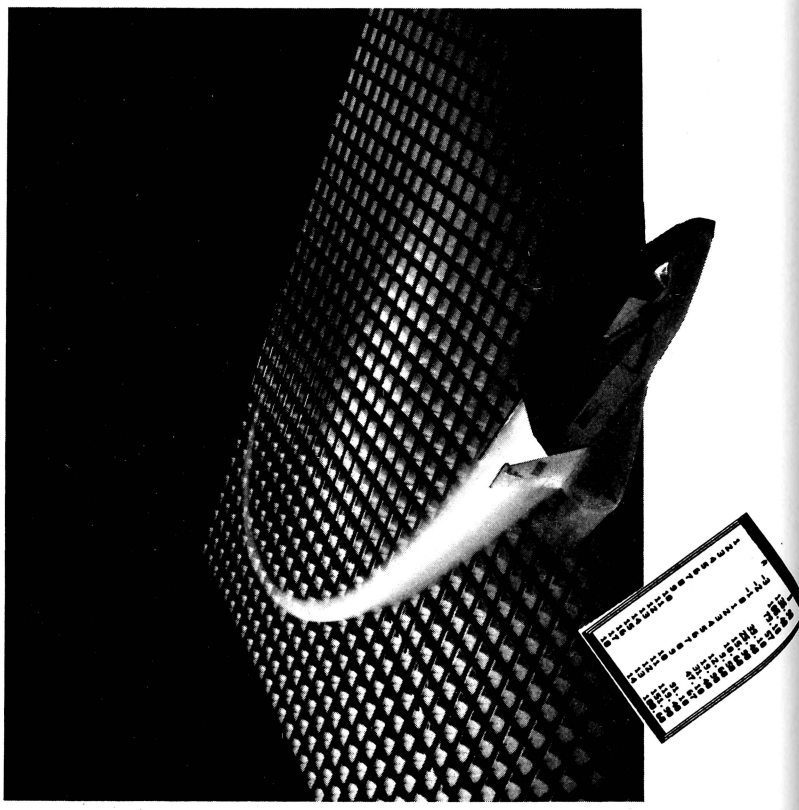
SET DECIMALS must be set higher than zero to force the output into decimal notation. Note also that one less decimal place displays than is requested. This is true for all versions of dBASE III PLUS.



Regression Analysis

A Framework II program for performing linear, multiple, and curvilinear regressions.

By Kenneth N. Getz and Chuck Litzell



Regression analysis is a statistical procedure that examines the relationships between variables. Two variables are related when changes in one variable affect the other. For example, businesses are interested in knowing how their advertising dollars will affect sales dollars. We might guess that there is a positive relationship between the two. As more dollars are invested in advertising, sales increase. Regression analysis produces an estimating equation from a set of sample observations. Given a value for one variable, the equation can be used to predict or estimate the value of the other variable. There are two types of variables: independent and dependent. The value of the dependent variable is said to depend on the value of the independent variable. The distinction is important; we could predict sales volume for various levels of advertising, but we should not attempt to predict advertising expenditures from sales volumes. There is a clear cause and effect relationship which should be maintained throughout the analysis.

In regression analysis, there is always one dependent variable, the one calculated by the regression equation. When there is more than one independent variable in an analysis, it is called multiple regression.

The programs in this article perform three types of regression analyses:

- simple linear regression

- multiple linear regression (two independent variables)
- curvilinear regression (quadratic)

Simple Linear Regression

Simple linear regression is concerned with one independent variable and one dependent variable. By substituting values for the independent variable in the equation, the dependent variable's value can be estimated. When the two variables are plotted against each other on an X-Y graph, the result is a straight line, thus the name linear regression. It is customary to plot the independent variable on the X-axis and the dependent variable on the Y-axis. The equation produced by linear regression is of the form

$y = a + bx$

where x is the independent variable, and y is the calculated dependent variable. a is called the constant term or y-intercept. It is the value of y when x is equal to zero. A graph of the line will cross the y-axis at the y-intercept.

b is called the coefficient or slope. It is the amount of change in y for a change of 1 in x. The slope defines the steepness of the graphed line. If the slope is positive, the line moves up as x increases and the relationship is said to be positive, as in our advertising-sales example. A negative slope is an indication of an inverse relationship. As x increases, y decreases.

The Fred program which performs the Simple Linear Regression is adapted from a program example in *Framework II Advanced Topics*. Note there are two typographical errors in the program, which begins on page 1-103. In the fourth line of the program, "@itemf1" should be replaced with "@item1". The first line at the top of page 1-104 should read

```
t3 := sumxy - (sumy * avgx),
```

Multiple Linear Regression

The second regression model provided is multiple linear regression on two independent variables. The independent variables are x_1 and x_2 in the equation

$y = a + b_1x_1 + b_2x_2$

The constant term, a, remains, but each of the independent variables has its own coefficient. The graph of this equation is more difficult to visualize, since it is three dimensional, but the principles are the same. An example of multiple linear regression would be if we calculated an equation to estimate sales given dollars spent on media advertising and dollars spent on direct sales contacts. The regression analysis would produce separate weights (coefficients) for advertising and direct sales. Direct sales may contribute more to increased sales volume than the advertising, for instance.

Curvilinear Regression

The curvilinear regression model is really a combination of the other two. As in simple linear regression, there is one independent variable and one dependent variable. The equation is

$y = a + bx + cx^2$

The inclusion of the x^2 term makes this a quadratic equation. Since there is just one independent variable, the quadratic equation can be graphed on an X-Y graph. A quadratic equation graphs as a vertical parabola, a curve that is open on the top or bottom (Figure 1). If c is positive, the opening is on top. If c is negative, the opening is on the bottom. Again, the constant term, a, is the value of y when the curve crosses the Y-axis, which occurs when x is equal to 0.

Quadratic equations are common in the business world. A quadratic relationship often exists between production quantities and costs. As quantity, the independent variable, increases, costs decrease, until scarcity of materials and similar factors begin to increase costs again.

The algorithm used to calculate the quadratic equation in the program is identical to the algorithm that calculates the two-variable multiple regression equation. This is valid because the quadratic equation can be transformed to the multiple linear equation by substituting x^2 in place of x_2, and c in place of b_2 in the multiple regression equation.

Creating Regress.FW2

1. To create Regress.FW2, first create an empty word frame, and label it Regress. Toggle **Frames:Number Labels** on. Inside of Regress, create an outline that looks like this:

```
Regress
   1     Select Regression Type
         1.1 Linear Regression
         1.2 Multiple Linear Regression
         1.3 Quadratic Regression
         1.4  Quit
   2     Report Options
         2.1  Print
         2.2  Quit
   3     Linear
   4     Multi Linear
   5  Quadratic
   6     Get range
   7     Report
   8     N
   9  Slope1
   10 Slope2
   11 Intercept
   12 Standard Error
   13 Coefficient
   14 Regression Report
```

2. In the contents area of the frame labeled 1.1 Linear Regression enter "Linear Regression with one independent variable."
3. In the contents area of the frame labeled 1.2 Multiple Linear Regression enter "Linear Regression with two independent variables."
4. In the contents area of the frame labeled 1.3 Quadratic Regression enter "Quadratic Regression with one independent variable."
5. In the contents area of the frame labeled 1.4 Quit enter "Quit Regression Program."
6. In the contents area of the frame labeled 2.1 Print enter "Print the Regression Summary Report."
7. In the contents area of the frame labeled 2.2 Quit enter "Return control to previous menu."
8. Enter the program code at the end of this article into the formula areas of the corresponding frames:

```
Regress
1.1 Linear Regression
1.2 Multiple Linear Regression
1.3 Quadratic Regression
1.4 Quit
2.1 Print
2.2 Quit
3   Linear
4   Multi Linear
5   Quadratic
6   Get range
7   Report
```

Using Regress

Before using Regress, you must have already entered your sample data observations into a spreadsheet or database. There must be a single region which holds the observations for the dependent (y) variable, and one or two regions to hold the corresponding independent observations. In each case, the regions must have the same number of cells. Each cell in the regions must hold a numeric value; there can be no empty cells.

Load Regress.FW2 to the desktop, highlight it and **RECALC (F5)**. Regress displays a lightbar menu at the bottom of the screen. Select one of the three regression models and press **Return**.

Next, Regress prompts for the (first) independent (x) region. You can enter the region directly or use cursor pointing to highlight it. If the region is a database field, enter the database name, a period, and the field name. For example, if the independent data were in a field named "Advertising Expense" in a database frame labeled "Market Study", you would enter

```
Market Study.Advertising Expense
```

If the data were in a region of cells in a spreadsheet, you would enter the region as follows:

```
Market Study.A2:Market Study.A50
```

If you want to use cursor pointing to select the region, begin by pressing **Uparrow**. Then navigate to the first cell in the region, press **EXTEND SELECT (F6)**, navigate to the last cell in the region and press **Return**. Press **Return** again to end cursor pointing and once more to enter the region into the program.

Note: When using cursor pointing to select a database field, simply highlight the field name in the database and press **Return**.

If you are performing a Multiple Linear Regression, Regress will ask you to identify the region which holds the sample observations for the second independent variable. Enter the region in the same way you entered the first.

Finally, you are asked to enter the region which holds the dependent (y) observations. Again, type the region as described above, or use cursor pointing to highlight it.

Regress then begins the regression procedure. The program uses Fred's region-walking procedures to access the data, so if there are many data observations, this make take a while.

When the calculations have been completed, a Regression Summary is displayed. At the bottom of the screen, a menu with two choices, "Print" and "Quit," is displayed. If you want to print the Regression Summary select "Print," otherwise, select "Quit."

Interpreting the Regression Summary

The Regression Summary reports the statistics calculated from your sample data. Following are explanations of these statistics.

Number of Observations (N)

This is the number of data observations included in the regression analysis.

Sample r-square (r^2)

The r^2 statistic is called the coefficient of correlation. It is interpreted as the portion of the variance in the sample y values that can be explained by the relationship between the independent and dependent values. r^2 varies from 0 to 1.0. A 1.0 value indicates that every data observation fit the calculated regression equation perfectly. This is rarely the case, but a high value suggests the equation is a "good fit" to the sample data. A low value indicates that the calculated equation does not fit the sample data very well. This may happen if you perform a linear regression on data that is curvilinear, or vice versa. It could also mean that the

relationship between the variables is weak, or that a different regression model should be used. This statistic, and the next, are provided to help you judge the reliability of the regression equation in calculating estimates of the dependent variable.

Sample Standard Error (S_e)

The Sample Standard Error statistic is used to calculate confidence intervals for estimated dependent variables. This calculation is quite complex, and we won't attempt to explain it here. However, a useful "rule of thumb" can be derived with S_e. About 68% of the y-estimates calculated with the regression equation will be within one S_e of their actual values. Obviously, if S_e is large, the estimates derived from the equation will not be very valuable.

Constant Term (a)

This is the y-intercept in the regression equation.

Variable 1 Coefficient (b)

The coefficient of the first independent variable. In Simple Linear Regression, this is the slope of the regression line.

Variable 2 Coefficient (c)

This statistic does not appear on the Regression Summary of a Simple Linear Regression. In Multiple Regression, it is the coefficient of the second independent variable. In the Quadratic Regression model, it is the coefficient of the x^2 term.

Forming the Regression Equation

The regression equation is easily constructed from the data on the Regression Summary. If the Simple Linear Regression model was used, the equation is

$y = a + bx$

Just substitute the values for a and b.

If the Multiple Linear Regression model was used, substitute the values for a, b and c in

$y = a + bx_1 + cx_2$

where x_1 is the first independent variable and x_2 is the second.

Finally, the Quadratic Regression equation can be formed by substituting the values for a, b, and c in

$y = a + bx + cx^2$

Regress

```
;Regress.FW2
; by Chuck Litzell and Kenneth N. Getz
; January 1, 1988
;
; Regression Analysis
;
; Call main menu
;
@menu(Regress.Select Regression Type)
```

```
Regress.Select Regression Type.Linear Regression
;
; Regress a single independent variable against a dependent variable.
;
@local(range1, range2, func),
range1 := @Get range("Range containing X observations."),
range2 := @Get range("Range containing Y observations."),
@eraseprompt,
@prompt("Working ... Please Wait"),
func  := "@Regress.Linear(" & range1 & ", " & range2 & ")",
@if(@func <> -1,
    @Regress.Report(1)
)
```

```
Regress.Select Regression Type.Multiple Linear Regression
;
; Regress two independent variables against a dependent variable.
;
@local(range1, range2, range3, func),
range1 := @Get range("Range containing X1 observations."),
range2 := @Get range("Range containing X2 observations."),
range3 := @Get range("Range containing Y observations."),
@eraseprompt,
@prompt("Working ... Please Wait"),
func   := "@Regress.Multi Linear(" & range1 & "," &
                    range2 & "," & range3 & ")",
@if(@func <> -1,
    @Regress.Report(2)
)
```

```
Regress.Select Regression Type.Quadratic Regression
;
; Regress a single (quadratic) independent variable against
; a dependent variable.
;
@local(range1, range2, func),
range1 := @Get range("Range containing X observations."),
range2 := @Get range("Range containing Y observations."),
@eraseprompt,
@prompt("Working ... Please Wait"),
func  := "@Regress.Quadratic(" & range1 & ", " & range2 & ")",
@if(@func <> -1,
    @Regress.Report(3)
)
```

```
Regress.Select Regression Type.Quit
; Return control to desk top
@quitmenu
```

```
Regress.Report Options.Print
; Regress.Report Options.Print
;
;  Print the Regression Report
;
@performkeys("{Ctrl-P}B")
```

```
Regress.Report Options.Quit
; Regress.Report Options.Quit
;
;   Return to the Regress menu.
;
@quitmenu
```

```
Regress.Linear
; Regress.Linear.FW2
;   Adapted from Advanced Topics, Framework II, pp. 1-103 to 1-104.
;
;    @Linear(range1, range2)
;            range1:  Observations for independent variable (x)
;            range2:  Observations for dependent variable  (y)
```

```
;
@local(sumx, sumy, sumxx, sumyy, sumxy),
@local(t1, t2),
;
sumx  := 0,
sumy  := 0,
sumxx := 0,
sumyy := 0,
sumxy := 0,
N     := 0,
;
; Walk through the X and Y regions, accumulating sums.
;
@while((t1 := @get(@item1)) <> #NULL!,
    t2    := @get(@item2),
    sumx  := sumx + t1,
    sumy  := sumy + t2,
    sumxx := sumxx + (t1 * t1),
    sumyy := sumyy + (t2 * t2),
    sumxy := sumxy + (t1 * t2),
    N     := N + 1,
    @next(@item1),
    @next(@item2)
),
;
; Compute Y-intercept and slope.
;
Slope1    := (N * sumxy - sumx * sumy) /
                  (N * sumxx - sumx * sumx),
Intercept := (sumy - Slope1 * sumx) / N,
Standard Error :=
    @sqrt(
        (sumyy - Intercept * sumy - Slope1 * sumxy) /
                          (n - 2)
    ),
Coefficient :=
             ((N * sumxy - sumx * sumy) ^ 2) /
    ((N * sumxx - sumx * sumx) * (N * sumyy - sumy * sumy)),
@return(0)
```

```
Regress.Multi Linear
; Regress.Multi Linear
;
;   Linear Regression with 1 dependent variable and 2 independent
;   variables.
;
;   @Multi Linear(range1, range2, range3)
;
;            range1:  Observations for 1st independent variable (x1)
;            range2:  Observations for 2nd independent variable (x2)
;            range3:  Observations for dependent variable     (y)
;
; Variables to hold sums.
@local(sumx1, sumx2, sumy, sumx1y, sumx2y),
@local(sumx1x2, sumx12, sumx22, sumy2),
;
; Variables to hold intermediate calculations.
@local(compA, compB, compC, compD, compE, compF),
;
; Variables to hold working values from regions.
```

```
@local(x1, x2, y),
;
; Initialize local variables.
sumx1   := 0,
sumx2   := 0,
sumy    := 0,
sumx1y  := 0,
sumx2y  := 0,
sumx1x2 := 0,
sumx12  := 0,
sumx22  := 0,
sumy2   := 0,
;
; Initialize number of observations.
N := 0,
;
; Walk through the three ranges, accumulating sums.
;
@while((x1 := @get(@item1)) <> #NULL!,
    x2     := @get(@item2),
    y      := @get(@item3),
    sumx1  := sumx1 + x1,
    sumx2  := sumx2 + x2,
    sumy   := sumy + y,
    sumx1y := sumx1y + (y * x1),
    sumx2y := sumx2y + (y * x2),
    sumx1x2:= sumx1x2 + (x1 * x2),
    sumx12 := sumx12 + (x1 * x1),
    sumx22 := sumx22 + (x2 * x2),
    sumy2  := sumy2 + (y  * y),
    N      := N + 1,
    @next(@item1),
    @next(@item2),
    @next(@item3)
),
;
; Intermediate calculations
;
compA := N * sumx1y - sumx1 * sumy,
compB := N * sumx22  - sumx2 * sumx2,
compC := N * sumx1x2 - sumx1 * sumx2,
compD := N * sumx2y - sumx2 * sumy,
compE := N * sumx12 - sumx1 * sumx1,
compF := compE * compB - compC * compC,
@if(compF = 0,
    @list(
        @eraseprompt,
        @beep,
        @prompt("Divide by zero error. " &
                "Formula not valid for this data.  Press a key."),
        @nextkey,
        @return(-1)
    )
),
; Calculate Slopes.
Slope1 := (compA * compB - compC * compD) / compF,
Slope2 := (compD * compE - compA * compC) / compF,
;
; and the Intercept
;
Intercept := (sumy - Slope1 * sumx1 - Slope2 * sumx2) / N,
```

```
;
; and the Standard Error
;
Standard error :=
     @sqrt(
          (sumy2 - Intercept * sumy - Slope1 * sumx1y - Slope2 * sumx2y) /
          (n - 3)
     ),
; and the Coefficient of Multiple Determination
;
Coefficient :=
     (N * (Intercept * sumy + Slope1 * sumx1y + Slope2 * sumx2y) -
               sumy * sumy) / (N * sumy2  - sumy * sumy),
@return(0)
```

Regress.Quadratic

```
; Regress.Quadratic
;
;    Quadratic Regression with 1 dependent variable and 1 independent
;    variable.  Quadratic uses the same formula as the Multi Linear
;    except that the second independent variable is formed by
;    squaring the first.
;
;    @Quadratic(range1, range2)
;
;              range1:  Observations for independent variable (x)
;              range2:  Observations for dependent variable  (y)
;
; Variables to hold sums.
@local(sumx1, sumx2, sumy, sumx1y, sumx2y),
@local(sumx1x2, sumx12, sumx22, sumy2),
;
; Variables to hold intermediate calculations.
@local(compA, compB, compC, compD, compE, compF),
;
; Variables to hold working values from regions.
@local(x1, x2, y),
;
; Initialize local variables.
sumx1    := 0,
sumx2    := 0,
sumy     := 0,
sumx1y   := 0,
sumx2y   := 0,
sumx1x2  := 0,
sumx12   := 0,
sumx22   := 0,
sumy2    := 0,
;
; Initialize number of observations.
N := 0,
;
; Walk through the three ranges, accumulating sums.
;
@while((x1 := @get(@item1)) <> #NULL!,
;
; Square x1 to arrive at x2.
;
     x2      := x1 * x1,
```

```
    y      := @get(@item2),
    sumx1  := sumx1 + x1,
    sumx2  := sumx2 + x2,
    sumy   := sumy + y,
    sumx1y := sumx1y + (y * x1),
    sumx2y := sumx2y + (y * x2),
    sumx1x2:= sumx1x2 + (x1 * x2),
    sumx12 := sumx12 + (x1 * x1),
    sumx22 := sumx22 + (x2 * x2),
    sumy2  := sumy2 + (y  * y),
    N      := N + 1,
    @next(@item1),
    @next(@item2)
),
;
; Intermediate calculations
;
compA := N * sumx1y - sumx1 * sumy,
compB := N * sumx22  - sumx2 * sumx2,
compC := N * sumx1x2 - sumx1 * sumx2,
compD := N * sumx2y - sumx2 * sumy,
compE := N * sumx12 - sumx1 * sumx1,
compF := compE * compB - compC * compC,
@if(compF = 0,
    @list(
        @eraseprompt,
        @beep,
        @prompt("Divide by zero error.  " &
                    "Formula not valid for this data.  Press a key."),
        @nextkey,
        @return(-1)
    )
),
; Calculate Slopes.
Slope1 := (compA * compB - compC * compD) / compF,
Slope2 := (compD * compE - compA * compC) / compF,
;
; and the Intercept
;
Intercept := (sumy - Slope1 * sumx1 - Slope2 * sumx2) / N,
;
; and the Standard Error
;
Standard error :=
    @sqrt(
        (sumy2 - Intercept * sumy - Slope1 * sumx1y - Slope2 * sumx2y) /
        (n - 3)
    ),
; and the Coefficient of Multiple Determination
;
Coefficient :=
    (N * (Intercept * sumy + Slope1 * sumx1y + Slope2 * sumx2y) -
            sumy * sumy) / (N * sumy2  - sumy * sumy),
@return(0)

Regress.Get range

; Regress.Get range
;
;   Input range of spreadsheet or database cells for regression.
;   Cursor pointing is allowed.
```

```
;
;    @Get range(prompt)
;
;
@local(range, test, home),

home := @setselection,

@while(#TRUE,
     range := @inputline(@item1 & " (You may use cursor pointing)", "",
                                 #YES, #YES, #YES),
     test  := "@get(" & range & ")",
     @if(@not(@iserr(@test)),
          @list(
               test := "@reset(" & range & ")",
               @test,
               @setselection(home),
               @return(range)
          ),
          @list(
               @eraseprompt,
               @prompt("Error in Range Specification. Press a key."),
               @beep,
               @nextkey,
          )
     )
)
```

```
; Regress.Report
;
; Make Printable Regression Analysis Report, and Display it.
;
Regress.Regression Report := " REGRESSION SUMMARY: " &
     @choose(@item1,
          "SIMPLE LINEAR REGRESSION",
          "MULTIPLE LINEAR REGRESSION",
          "CURVILINEAR (QUADRATIC) REGRESSION"
     ) & "
     Number of Observations ..........(N)... " & @integer(N) & "
     Sample r-square .................(r2).. " & @decimal(Coefficient, 4) & "
     Sample Standard Error ...........(Se).. " & @decimal(Standard Error, 4) & "
     Constant Term ...................(a)... " & @decimal(Intercept, 4) & "
     Variable 1 Coefficient ..........(b)... " & @decimal(Slope1, 4) &
@if(@or(@item1 = 2, @item1 = 3), "
     Variable 2 Coefficient ..........(c)... " & @decimal(Slope2, 4) & "
"
) & "
",
@setselection("Regress.Regression Report"),
@performkeys("{F9}"),
@menu(Regress.Report Options),
@echo(#OFF),
@performkeys("{Ctrl-D}C"),
@echo(#ON)
```

Function Prototyping

A Development Tool for FRED Programmers

By Ken Getz and Chuck Litzell

A prototype is a working model of a final design. Using prototypes in product development is a common practice in many industries. The prototype is built, and then tested and modified until it performs to specifications.

Many programmers use a similar technique when debugging their programs. When a new function is needed for a program, they develop the function independently of the program where it will be used. Only when the function is tested and working is it added to the program. This approach to programming is especially appropriate for Fred programmers. The Framework II environment is well-suited to developing prototypes, and Fred programs can become sufficiently complicated to warrant using them.

Proto.FW2 is a Fred program that implements prototype development in Framework II. The prototype expression is developed outside of your application program. This protects the work you've already accomplished and speeds development since you don't have to run your program from the beginning for each test. The application's data environment is simulated by establishing test variables to be used by the prototype.

Using Proto

In the spirit of "learn by doing," the following is a sample Proto session which demonstrates how Proto can be used to develop a rather complex expression.

The expression we will develop reformats a database record into a comma-delimited string — the format that Framework II produces when exporting to dBASE. The database has five fields: Company, Address, City, State, and Zip Code. The string we want the expression to produce will look like

```
"Ashton-Tate","20101 Hamilton Avenue","Torrance","CA","90502"
```

To begin, load Proto to the desktop, highlight it, and **RECALC (F5)**. Proto's menu is displayed at the bottom of the screen. We need some data to work with, so highlight "Set local variables" and press **Return**.

Proto prompts, "Enter a variable name (press {Return} to Quit)." Type Company and press **Return**.

Next Proto instructs, "Enter Company's Value." Enter "Ashton-Tate" and press **Return**. Make sure you type the quotes; Proto requires that literal string values be entered with quotes when setting up local

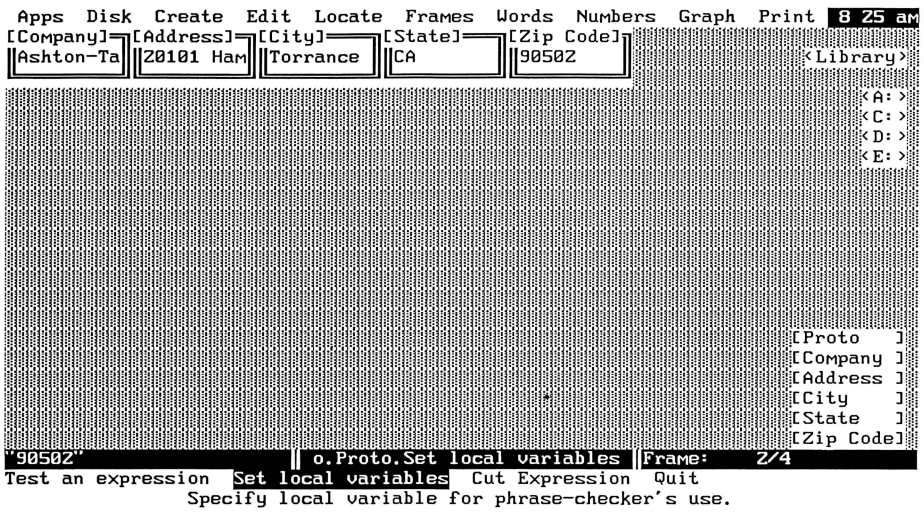

Apps Disk Create Edit Locate Frames Words Numbers Graph Print 8:25 am
[Company] [Address] [City] [State] [Zip Code]
Ashton-Ta 20101 Ham Torrance CA 90502
<Library>
<A:>
<C:>
<D:>
<E:>
[Proto]
[Company]
[Address]
[City]
[State]
[Zip Code]
"90502" o.Proto.Set local variables Frame: 2/4
Test an expression Set local variables Cut Expression Quit
Specify local variable for phrase-checker's use.

Figure 1

variables. If you enter an invalid expression when the program is asking for a value, it will beep and ask you to enter the value again.

Enter the next four variables in the same way:

```
Address:"20101 Hamilton Avenue"
City:    "Torrance"
State:     "CA"
Zip Code:  "90502"
```

When you're through, press **Return** to quit "Set local variables." Your screen should look like Figure 1.

Next, you can begin creating the prototype. Select "Test an expression" from the menu. This places two frames, Expression and Evaluates to, at the bottom of your screen. Proto prompts, "Type your command line (including extra quotes)."

The first thing in the delimited string will be a literal quotation mark. In Fred, a literal quote is written with two consecutive quotation marks within a pair of quotes. That's a total of four quotation marks. Enter the four quotes and press **Return**.

Proto displays the four quotes, as you typed them, in the Expression frame. In the Evaluates to frame, the translated value of the four quotes is displayed – a single quote.

The next item in the delimited string is the Company, which should follow immediately after the opening quotation. Select "Test an expression" again, and press **End** to move the cursor past the quotes. Enter

```
& Company
```

and press **Return**. Your screen should look like Figure 2.

Next, select "Test an expression" again, and press **End** to move to the end of your previous entry. We want to add a single quotation mark, a comma, and another quotation mark after Ashton-Tate. To do this with Fred, we need two quotes, a comma, and two more quotes, all surrounded by a set of quotes. Enter

```
& ""","""
```

and press **Return**. So far so good. Because we're building a prototype, we can build the expression bit-by-bit, making sure everything is OK before we go on. Let's take a bigger step this time. Add

```
& Address & """,""" & City &
"""," & State & ""","""
```

to the expression. It gets easier, since we've already found a format that works. To complete the expression, we need to add the Zip Code and terminate it with a single quote. That will require a pair of quotes surrounded by another pair of quotes. Add

```
& Zip Code & """"
```

to your expression. If you've done everything correctly, your screen should look like Figure 3.

Now that you've developed the prototype expression, you can Cut the expression to the Cut Paste Buffer.

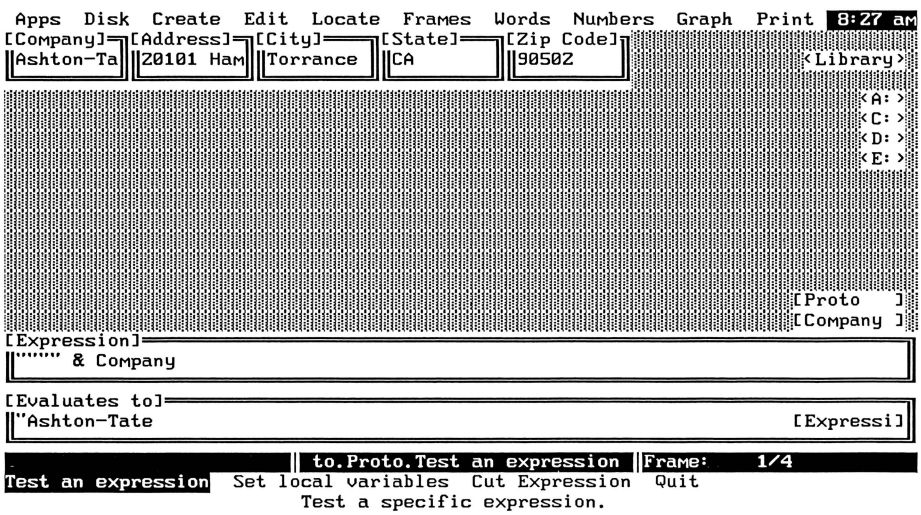

Apps Disk Create Edit Locate Frames Words Numbers Graph Print 8:27 am
[Company] [Address] [City] [State] [Zip Code]
Ashton-Ta 20101 Ham Torrance CA 90502
<Library>
<A:>
<C:>
<D:>
<E:>
[Proto]
[Company]
[Expression]
"""" & Company
[Evaluates to]
"Ashton-Tate [Expressi]
to.Proto.Test an expression Frame: 1/4
Test an expression Set local variables Cut Expression Quit
Test a specific expression.

Figure 2

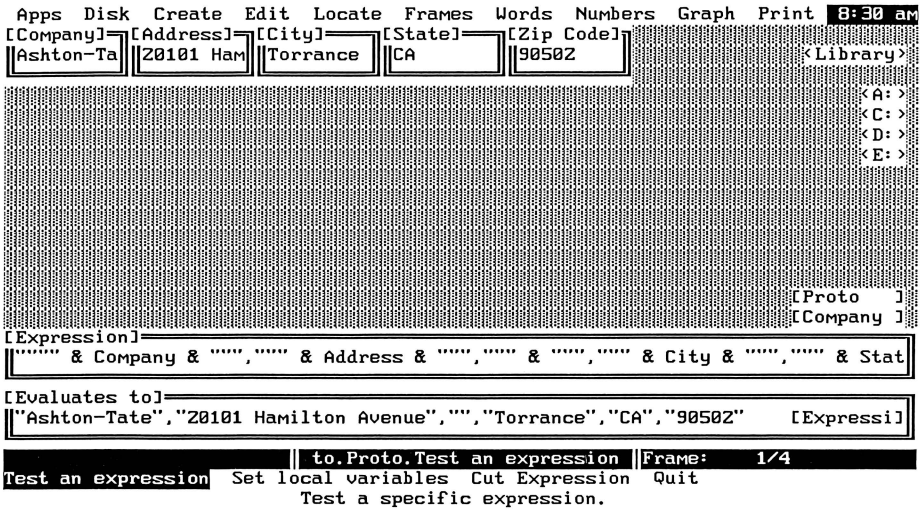

Apps Disk Create Edit Locate Frames Words Numbers Graph Print 8:30 am
[Company] [Address] [City] [State] [Zip Code]
Ashton-Ta 20101 Ham Torrance CA 90502
<Library>
<A:>
<C:>
<D:>
<E:>
[Proto]
[Company]
[Expression]
"""" & Company & """","""" & Address & """","""" & """","""" & City & """","""" & Stat
[Evaluates to]
"Ashton-Tate","20101 Hamilton Avenue","","Torrance","CA","90502" [Expressi]
to.Proto.Test an expression Frame: 1/4
Test an expression Set local variables Cut Expression Quit
Test a specific expression.

figure 3

Select "Cut expression" from Proto's menu. This selection simply executes the {Alt-F7} macro in your Library. Finally, select "Quit" to leave Proto.

If you were working with a real program, you would navigate to the place in your program where you wanted to use the expression you developed, and press **Alt-F8** to paste the expression into your program.

Notes and Hints

If you enter an expression that generates an error when evaluated, Proto displays "Error in expression" in the frame labeled Evaluates to. You can edit the expression until it evaluates correctly.

If your expression is long, or you want to enter it on multiple lines, ZOOM (F9) when Proto prompts for the expression. Enter your expression, then press Uplevel to return to the Proto menu.

You can change an existing local variable's value by selecting "Set local variables" from the menu, and entering that variable's name again. The new value you enter will replace the existing value.

The values you enter for local variables are placed in the variable's formula area and then recalculated. This means you can enter complex formulas which use Fred functions or other, existing variables.

It is not possible to create an uninitialized local variable. Enter a zero or a space (between quotes) if you don't have an actual value to enter.

Creating Proto

To create Proto.FW2, follow these steps:

1. Create an empty frame on the desktop and label it Proto.

2. Inside Proto, create subframes and label them so the frame looks like this:

```
- Proto
  - Main
  - Proto
    - Test an expression
    - Set local variables
    - Cut expression
    - Quit
  - Lastrow
  - Lastcol
  - Islocal
```

3. Highlight the containing frame, and toggle **Frames:Number Labels** on.

4. In the contents area of the frame labeled Test an expression, enter "Test a specific expression."

5. In the contents area of the frame labeled Set local variables, enter "Specify local variable for phrase-checker's use."

6. In the contents area of the frame labeled Cut expression, enter "Cut expression to Cut Paste Buffer."

7. In the contents area of the frame labeled Quit, enter "Quit the program."

8. Into the formula areas of the following frames, enter the corresponding program code from the end of this article:

```
Proto
Proto.Main
Proto.Proto.Test an expression
Proto.Proto.Set local variables
Proto.Proto.Cut expression
Proto.Proto.Quit
```

Creating Sizeto

Sizeto is a user-defined function that creates frames on the desktop with locations and dimensions passed as parameters. Sizeto was originally published in the June 1987 issue of ***TechNotes***. Proto uses Sizeto to create the frames it uses to store and evaluate expressions and local variables. If you have already created Sizeto, all you need to do is make sure it's in your Library or on the desktop when you use Proto.

If you haven't created Sizeto, follow these steps:

1. Create an empty frame on the desktop and label it Sizeto.

2. Inside of Sizeto, create two subframes. Label them Max Cols and Max Rows.

3. In the formula area of Sizeto, enter the program code from the end of this article.

4. Move Sizeto into the User-Defined Functions area of your Library.

Proto

```
;Proto.FW2
;   by Kenneth N. Getz
;   January 1, 1988
;
; This program allows creating prototypes of complicated FRED expressions
; which include variables and multiple quotes. To run, load this frame
; to the desktop and RECALC (F5).
;
@Proto.Main
```

Proto.Main

```
; Proto.Main
;
; Initialize the global variables.
;
Lastrow := 1,
Lastcol := 1,
Islocal := #FALSE,
;
; Clean up the Screen.
;
@echo(#OFF),
@performkeys("{Ctrl-D}C"),
@echo(#ON),
;
; Call the menu.
;
@menu(Proto)
```

Proto.Proto.Test an expression

```
; Proto.Test an expression
;
@echo(#OFF),
@local(temp),
;
; If there are no previously declared local variables, clean up the
; desktop. See if "Expression" already exists. If not, create and
; size it. Make sure it's open and the right margin is set to 255.
;
@if(@iserr(@setselection("Expression")),
    @list(
        @if(@not(Islocal),
            @performkeys("{Ctrl-D}C")
        ),
        @Sizeto("Expression", 16, 1, 18, 80),
        Expression := " "
    )
),
@performkeys("{Ctrl-Dnlevel}{Ctrl-Uplevel}{Ctrl-W}R255{Return}"),
;
; Make sure "Evaluates to" exists. If not, create it and size it.
; Make sure it's open and the right margin is set to 255.
;
@if(@iserr(@setselection("Evaluates to")),
    @Sizeto("Evaluates to", 19, 1, 21, 80)
),
@list(
@performkeys("{Ctrl-Dnlevel}{Ctrl-Uplevel}{Ctrl-W}R255{Return}"),
```

```
Evaluates to := " ",
),
@echo(#ON),
@setselection("Expression"),
;
; Get the command line, defaulting to the last entry.
;
temp := @inputline("Type your command line (including extra quotes)",
                    Expression, #NO, #NO, #YES),
@echo(#ON),
;
; If the expression contains an error, say so. Then store the
; expression for the next time around, and evaluate it.
;
Evaluates to := @if(@iserr(@temp),
                   "Error in expression.",
                    @temp
                ),
Expression := temp
```

Proto.Proto.Set local variables

```
; Proto.Set local variables
;
; This program asks for variable names and values and lays them out
; neatly on the desktop. If you need variables longer than 10 chars,
; you will need to make a few minor changes.
;
@local(fname, fvalue, temp, rows, cols, inlibrary, prompt),
;
; Indicate that there are now some local variables on the desktop.
;
Islocal  := #TRUE,
prompt   := "Enter a variable name (press {Return} for Menu)",
;
; The following are constants indicating the number of rows and
; columns in each variable. Adjust these if you need larger or
; smaller variables.
;
rows := 3,
cols := 11,
;
@while(#TRUE,
;
;    If the user presses {Return}, Quit.
;
    @if((fname := @inputline(prompt)) = #N/A!,
        @return(0)
    ),
;
;    If the new variable doesn't already exist, create a new frame.
;
    @echo(#OFF),
    @if(@or(@iserr(@setselection(fname)),
        (inlibrary := (@scan("Library", @setselection) <> #N/A!))),
        @list(
            @Sizeto(fname, Lastrow, Lastcol, Lastrow + rows - 1,
                    Lastcol + cols - 1),
            @setselection(fname),
            fname   := @setselection,
            Lastcol := Lastcol + cols,
```

```
            @if(inlibrary,
                @list(
                    @setselection("Library"),
                    @performkeys("{Return}"),
                    @setselection(fname)
                )
            )
        )
    ),
    @echo(#ON),
;
;   Correct screen positions for next variable.
;
    @if(@or(Lastcol >= Max Cols - cols, Lastcol + cols > Max Cols),
        @list(
            Lastcol := 1,
            Lastrow := Lastrow + rows
        )
    ),
;
;   Value entered is placed into the formula area of the frame,
;   then recalculated to display the result.
;
    @while(
        @iserr(
            @list(
                fvalue := @inputline("Enter " & fname & "'s value"),
                temp   := "@execute(" & fname & "," & fvalue & ")",
                @temp
            )
        ),
        @list(
            @beep,
            @eraseprompt,
            @prompt("Error in Value. Press a key."),
            @nextkey
        )
    ),
  @performkeys("{F5}")
)
```

Proto.Proto.Cut expression

```
; Proto.Cut expression
;
; Cut Expression to Cut Paste Buffer.
;
@if(@iserr(@setselection("Expression")),
    @list(
        @eraseprompt,
        @beep,
        @prompt("There is no Expression to Cut. Press a key."),
        @nextkey
    ),
    @list(
        @performkeys("{Ctrl-Dnlevel}{Ctrl-Home}{F6}{Ctrl-End}{Return}" &
                     "{Alt-F7}{Esc}{Ctrl-Uplevel}"),
        @eraseprompt,
        @prompt("Expression has been cut to Cut Paste Buffer. Press a key."),
        @nextkey
    )
)
```

Proto.Proto.Quit

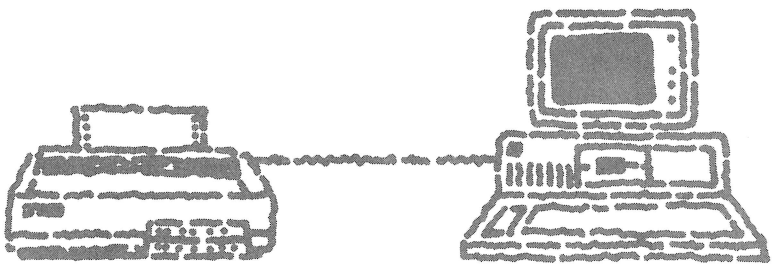
```
; Proto.Quit
;
; Return control to desktop.
;
@quitmenu
```

Sizeto

```
; Sizeto
;
;   by Kenneth N. Getz
;      June 1, 1987
;
; Used to create a frame at a specified location on the screen.
;
; @Sizeto(<frame name>, <upper left row>, <upper left col>,
;             <lower right row>, <lower right col>)
;
; Returns 0 if frame formed OK, -1 if there was an error.
;
; Make sure we're at the desktop level.
@performkeys("{Ctrl-Uplevel}"),
;
; Create the new frame and zoom to full screen, storing away the maximum
; screen coordinates. Verify that the parameters passed conform to the
; possible position values.
;
@performkeys("{Ctrl-C}E{F4}{F4}"),
Max rows := @panel2,
Max cols := @panel1,
@if(@or(@item2 < 1, @item3 < 1, @item4 > Max Rows + 2, @item5 > Max Cols + 2),
     @list(
          @performkeys("{Return}{Del}"),
          @return(-1)
     )
),
; Perform DRAG (F3) to move upper left corner.
@performkeys("{F3}"),
@performkeys(@rept("{Ctrl-Dnarrow}", @int((@item2 - 1)/4)) &
              @rept("{Dnarrow}", @mod(@item2 - 1, 4)) &
              @rept("{Ctrl-Rightarrow}", @int((@item3 - 1)/8)) &
              @rept("{Rightarrow}", @mod(@item3 - 1, 8)) &
              "{Return}"
),
;
; Perform SIZE (F4) to move lower right corner.
@performkeys("{F4}" &
              @rept("{Ctrl-Uparrow}", @int((Max rows + 2 - @item4)/4)) &
              @rept("{Uparrow}", @mod(Max rows + 2 - @item4, 4)) &
              @rept("{Ctrl-Leftarrow}", @int((Max cols + 2 - @item5)/8)) &
              @rept("{Leftarrow}", @mod((Max cols + 2 - @item5), 8)) &
              "{Return}"
),
;
; Put the name in.
@performkeys(@item1 & "{Return}"),
@return(0)
```

Printer Driver Update

Because many new printer drivers are now available for Framework II, we present an updated table.



In the August 1986 issue of ***TechNotes***, the article "Framework II Printer Drivers," by Karen Robinson, provided a comprehensive discussion of the Framework II printer drivers that were then available. Since that time, numerous new printer drivers have been written for Framework II. The table included here is an up-to-date listing of all currently available printer drivers for Framework II.

The table lists the supported printers and print attributes available through Framework II. If the table indicates "Yes" for an attribute, Framework II will send the appropriate codes for that particular printer; otherwise, the alternate print style noted is used.

If your printer is not listed, use the generic driver. Graphs will print with text characters, bold will print as doublestrike, and underline and italics as underscores. If your printer emulates another, you may be able to use that driver; however, graphs may not print correctly. If your printer emulates another printer for which a driver is listed you may be able to use that driver; however, graphs may not print correctly.

You have several options for obtaining copies of new printer drivers. You can contact the Ashton-Tate Software Support Center or you can download directly from the Ashton-Tate BBS or Compuserve. For details on how to download programs, see "On-Line Services," page 71, in this issue of ***TechNotes.***

Framework II Printer Drivers

Manufacturer	Model	Driver	Bold	Italic	Underline	Graphs	Compressed	Letter Quality
AMT	AMT 250	AMT.PR (b)	Double	Yes	Dashed	Yes	No	Yes
	AMT 250 Color (4 color)	AMTCLR.PR (a) (b)	Double	Yes	Dashed	Yes	No	Yes
Anadex	6000	ANA6000.PR (b)	Double	Yes	Yes	Yes	Yes	Yes
	9625B (Silent Scribe)	ANA9625B.PR	Yes	Underline	Dashed	Yes	Yes	Yes
Brother	HR-15	HR15.PR	Yes	Underline	Yes	Yes	Yes	N/A
	HR-25	HR15.PR	Yes	Underline	Yes	Yes	Yes	N/A
Cannon	A2							
	In 150dpi graphics mode	CANON_A2.150 (b,d)	Yes	Yes	Yes	Yes	Yes	No
	In 300dpi graphics mode	CANON_A2.300 (b,d)	Yes	Yes	Yes	Yes	Yes	No
C. Itoh	1550A, B, BPI	CI1550A.PR	Yes	Underline	Yes	Yes	Yes	No
	1550S	CI1550S.PR	Yes	Yes	Yes	Yes	Yes	No
	1550SC	CI1550SC.PR (a)	Yes	Yes	Yes	Yes	Yes	No
	715	CI715_2.PR	Yes	Yes	Yes	Yes	Yes	Yes
	815 (P351 mode)	CI815.PR (b)	Yes	Yes	Yes	Yes	Yes	Yes
	8510 A, B, BPI	CI8510A.PR	Yes	Underline	Yes	Yes	Yes	No
	8510S	CI8510S.PR	Yes	Yes	Yes	Yes	Yes	No
	8510SC	CI8510SC.PR (a)	Yes	Yes	Yes	Yes	Yes	No
	3500 model 20	CI3520.PR	Yes	Yes	Yes	Yes	Yes	Yes
	ProwriterJr.							
	Standard graphs	DP_8050.PR	Yes	Underline	Yes	Yes	Yes	Yes
	High-resolution graphs	FX80_3.PR	Yes	Yes	Yes	Yes	Yes	Yes
DataSouth	180	DS180.PR	Double	Underline	Yes	Yes	Yes	No
	220	DS220.PR	Double	Underline	Dashed	Yes	Yes	Yes
DataProducts	P-80	DP_P80.PR	Double	Dashed	Dashed	Yes	Yes	Yes
	P-132	DP_P80.PR	Double	Dashed	Dashed	Yes	Yes	Yes
	SPG-8050, SPG-8070							
	(no color)	DP_8050.PR	Yes	Underline	Yes	Yes	Yes	Yes
	SPG-8050, SPG-8070							
	(4 color)	DP_8050C.PR(a)	Yes	Underline	Yes	Yes	Yes	Yes
Diablo	(Xerox) 630 API	DIAB630A.PR	Yes	Underline	Yes	Yes	No	N/A
Epson	MX-80	GENERIC.PR	Double	Dashed	Dashed	No	No	No
	MX-80 with Graftrax	MX80G.PR	Yes	Yes	Yes	Yes	Yes	Yes
	MX-80 with							
	Graftrax PLUS							
	Standard graphs	FX80_2.PR	Yes	Yes	Yes	Yes	Yes	Yes

Manufacturer	Model	Driver	Bold	Italic	Underline	Graphs	Compressed	Letter Quality
	High-resolution graphs	FX80_3.PR	Yes	Yes	Yes	Yes	Yes	Yes
	MX-100	GENERIC.PR	Double	Underline	Yes	No	No	No
	MX-100 with Graftrax	MX80G.PR	Yes	Yes	Yes	Yes	Yes	Yes
	MX-100 with Graftrax PLUS							
	Standard graphs	FX80_2.PR	Yes	Yes	Yes	Yes	Yes	Yes
	High-resolution graphs	FX80_3.PR	Yes	Yes	Yes	Yes	Yes	Yes
	RX-80, FX-80, FX-80 PLUS							
	Low-resolution graphs	FX80_1.PR	Yes	Yes	Yes	Yes	Yes	Yes
	Standard graphs	FX80_2.PR	Yes	Yes	Yes	Yes	Yes	Yes
	High-resolution graphs	FX80_3.PR	Yes	Yes	Yes	Yes	Yes	Yes
	RX-80/FX-80 with 8177	EPSN8177.PR	Double	Yes	Yes	Yes	Yes	Yes
	RX-100, FX-100, FX-100 PLUS							
	Standard graphs	FX80_2.PR	Yes	Yes	Yes	Yes	Yes	Yes
	High-resolution graphs	FX80_3.PR	Yes	Yes	Yes	Yes	Yes	Yes
	RX-100/FX-100 with 8177	EPSN8177.PR	Double	Yes	Yes	Yes	Yes	Yes
	FX-85							
	Low-resolution graphs	FX85_1.PR	Yes	Yes	Yes	Yes	Yes	Yes
	Standard graphs	FX85_2.PR	Yes	Yes	Yes	Yes	Yes	Yes
	High-resolution graphs	FX85_3.PR	Yes	Yes	Yes	Yes	Yes	Yes
	FX-86E							
	Low-resolution graphs	FX86E_1.PR (b)	Yes	Yes	Yes	Yes	Yes	Yes
	Standard graphs	FX86E_2.PR (b)	Yes	Yes	Yes	Yes	Yes	Yes
	High-resolution graphs	FX86E_3.PR (b)	Yes	Yes	Yes	Yes	Yes	Yes
	FX-185							
	Low-resolution graphs	FX85_1.PR	Yes	Yes	Yes	Yes	Yes	Yes
	Standard graphs	FX85_2.PR	Yes	Yes	Yes	Yes	Yes	Yes
	High-resolution graphs	FX85_3.PR	Yes	Yes	Yes	Yes	Yes	Yes
	LX-80	LX80.PR	Yes	Yes	Yes	Yes	Yes	Yes
	LX-90	LX80.PR	Yes	Yes	Yes	Yes	Yes	Yes
	DX-10	DIAB630A.PR	Yes	Underline	Yes	Yes	No	N/A
	DX-20	DIAB630A.PR	Yes	Underline	Yes	Yes	No	N/A
	DX-35	DIAB630A.PR	Yes	Underline	Yes	Yes	No	N/A
	JX-80 with 8177	EPSN8177.PR	Double	Yes	Yes	Yes	Yes	Yes
	JX-80 (8 color)	JX80_2.PR(a)	Yes	Yes	Yes	Yes	Yes	Yes
	LQ-1500							
	Standard graphs	LQ1500_1.PR(b)	Yes	Yes	Yes	Yes	Yes	Yes
	High-resolution graphs	LQ1500_2.PR	Yes	Yes	Yes	Yes	Yes	Yes
	SQ-2000	LQ1500.PR(b)	Yes	Yes	Yes	Yes	Yes	Yes
	GQ-3500							
	Standard graphs	GQ35_150.PR(b, e)	Yes	Underline	Yes	Yes	Yes	No
	High-resolution graphs	GQ35_300.PR(b, e)	Yes	Underline	Yes	Yes	Yes	No

continued

Manufacturer	Model	Driver	Bold	Italic	Underline	Graphs	Compressed	Letter Quality
Facit	4512 FAC	4512.PR	Yes	Underline	Yes	Yes	Yes	Yes
Fujitsu	DotMax 9f							
	Standard graphs	FX80_2.PR	Yes	Yes	Yes	Yes	Yes	Yes
	High-resolution graphs	FX80_3.PR	Yes	Yes	Yes	Yes	Yes	Yes
	DotMax 9i	IBMGP.PR	Yes	Underline	Yes	Yes	Yes	Yes
	DotMax 24i	FUJ24I.PR	Yes	Yes	Yes	Yes	Yes	Yes
	DotMax 24c (8 color)	FUJ24C.PR (a)	Yes	Yes	Yes	Yes	Yes	Yes
Generic Driver	Any printer not listed	GENERIC.PR	Double	Dashed	Dashed	No	No	No
HP	DeskJet (HP 2276)							
	In 150dpi graphics mode	HPDSK150.PR (b)	Yes	Yes	Yes	Yes	Yes	No
	In 300dpi graphics mode	HPDSK300.PR (b)	Yes	Yes	Yes	Yes	Yes	No
	ThinkJet	HPJET.PR	Yes	Underline	Yes	Yes	Yes	No
	LaserJet							
	In 100dpi graphics mode	HPLAS100.PR	Double	Yes	Yes	Yes	No	No
	In 150dpi graphics mode	HPLAS150.PR	Double	Yes	Yes	Yes	No	No
	In 300dpi graphics mode	HPLAS300.PR	Double	Yes	Yes	Yes	No	No
	Landscape	HPLASL.PR	Double	Underline	Yes	Yes	Yes	No
	QuietJet							
	Standard graphs	HPQJET1.PR (b)	Yes	Underline	Yes	Yes	Yes	Yes
	High-resolution graphs	HPQJET2.PR (b)	Yes	Underline	Yes	Yes	Yes	Yes
IBM	Graphics Printer	IBMGP.PR	Yes	Underline	Yes	Yes	Yes	Yes
	ProPrinter							
	Low-resolution graphs	IBMPRO_1.PR	Yes	Underline	Yes	Yes	Yes	Yes
	Standard graphs	IBMPRO_2.PR	Yes	Underline	Yes	Yes	Yes	Yes
	High-resolution graphs	IBMPRO_3.PR	Yes	Underline	Yes	Yes	Yes	Yes
	Color Jet Printer							
	(Monochrome)	IBMCJET2.PR	Yes	Underline	Yes	Yes	Yes	Yes
	QuietWriter							
	(model 1)	IBMQUIET.PR	Double	Underline	Yes	Yes	No	No
	Quietwriter							
	(model 2)							
	Low-resolution graphs	IBMQ2_1.PR (b)	Double	Underline	Yes	Yes	No	No
	Standard graphs	IBMQ2_2.PR (b)	Double	Underline	Yes	Yes	No	No
	High-resolution graphs	IBMQ2_3.PR (b)	Double	Underline	Yes	Yes	No	No
	Quietwriter							
	(model 3)							
	Low-resolution graphs	IBMQ3_1.PR (b)	Yes	Underline	Yes	Yes	Yes	Yes
	Standard graphs	IBMQ3_2.PR (b)	Yes	Underline	Yes	Yes	Yes	Yes
	High-resolution graphs	IBMQ3_3.PR (b)	Yes	Underline	Yes	Yes	Yes	Yes
	WheelPrinter	IBMWHEEL.PR	Double	Underline	Yes	Yes	Yes	N/A

Manufacturer	Model	Driver	Bold	Italic	Underline	Graphs	Compressed	Letter Quality
NEC	Pinwriter P2	IBMGP.PR	Yes	Underline	Yes	Yes	Yes	Yes
	Pinwriter P3	IBMGP.PR	Yes	Underline	Yes	Yes	Yes	Yes
	Pinwriter P5	NECP5.PR (b)	Yes	Yes	Yes	Yes	Yes	Yes
	Spinwriter 3550	NEC3550.PR	Double	Underline	Yes	Yes	Yes	N/A
	Spinwriter 7710	NEC7710.PR	Yes	Underline	Yes	Yes	No	N/A
	Spinwriter 8850	NEC8850.PR	Yes	Underline	Yes	Yes	Yes	N/A
Okidata	Microline 80	GENERIC.PR	Double	Underline	Yes	No	No	No
	Microline 82	GENERIC.PR	Double	Underline	Yes	No	No	No
	Microline 82A with Okigraph	OKI82AGR.PR	Double	Underline	Yes	Yes	Yes	No
	Microline 83A	GENERIC.PR	Double	Underline	Yes	No	No	No
	Microline 83A with Okigraph	OKI82AGR.PR	Double	Underline	Yes	Yes	Yes	No
	Microline 84	OKI84.PR(b)	Double	Underline	Yes	Yes	Yes	Yes
	Microline 182	OKI182.PR	Double	Underline	Yes	Yes	Yes	Yes
	182 Plug 'n Play	OKI182PN.PR	Yes	Underline	Yes	Yes	Yes	Yes
	Microline 183	OKI182.PR	Double	Underline	Yes	Yes	Yes	Yes
	183 Plug 'n Play	OKI182PN.PR	Yes	Underline	Yes	Yes	Yes	Yes
	Microline 92	OKI92_2.PR	Yes	Underline	Yes	Yes	Yes	Yes
	Microline 93	OKI92_2.PR	Yes	Underline	Yes	Yes	Yes	Yes
	Microline 192	OKI192_2.PR	Yes	Yes	Yes	Yes	Yes	Yes
	Microline 193	OKI192_2.PR	Yes	Yes	Yes	Yes	Yes	Yes
	92 or 192 with Plug 'n Play	OKI92PNP.PR	Yes	Underline	Yes	Yes	Yes	Yes
	93 or 193 with Plug 'n Play	OKI92PNP.PR	Yes	Underline	Yes	Yes	Yes	Yes
	Pacemark 2350	OKI2410.PR	Double	Underline	Yes	Yes	Yes	Yes
	Pacemark 2410	OKI2410.PR	Double	Underline	Yes	Yes	Yes	Yes
	Okimate-20	IBMGP.PR	Yes	Underline	Yes	Yes	Yes	Yes
Qume	Sprint 11 Plus	QUME11.PR	Yes	Underline	Yes	Yes	Yes	N/A
	LetterPro 20	QUME11.PR	Yes	Underline	Yes	Yes	Yes	N/A
Ricoh	6000							
	Portrait Orientation							
	100 dpi graphics mode	RIC6K10.PR (b)	Yes	Underline	Yes	Yes	Yes	No
	150 dpi graphics mode	RIC6K15.PR (b)	Yes	Underline	Yes	Yes	Yes	No
	300 dpi graphics mode	RIC6K30.PR (b)	Yes	Underline	Yes	Yes	Yes	No
	6000							
	Landscape Orientation							
	100 dpi graphics mode	RIC6K10L.PR (b, c)	Yes	Underline	Yes	Yes	Yes	No

continued

Manufacturer	Model	Driver	Bold	Italic	Underline	Graphs	Compressed	Letter Quality
	150 dpi graphics mode	RIC6K15L.PR (b, c)	Yes	Underline	Yes	Yes	Yes	No
	300 dpi graphics mode	RIC6K30L.PR (b, c)	Yes	Underline	Yes	Yes	Yes	No
Star Micronics	Gemini-10x	GEM10.PR	Yes	Yes	Yes	Yes	Yes	Yes
	Gemini-15x	GEM10.PR	Yes	Yes	Yes	Yes	Yes	Yes
TI	855	TI855.PR	Yes	Yes	Yes	Yes	Yes	Yes
	865	TI855.PR	Yes	Yes	Yes	Yes	Yes	Yes
Tandy	CGP220	CPG220.PR (b)	No	No	No	Yes	No	No
Toshiba	P1340	P1340.PR	Double	Underline	Yes	Yes	Yes	Yes
	P1351	P1340.PR	Double	Underline	Yes	Yes	Yes	Yes
	P321	P351.PR	Yes	Yes	Yes	Yes	Yes	Yes
	P341	P351.PR	Yes	Yes	Yes	Yes	Yes	Yes
	P351	P351.PR	Yes	Yes	Yes	Yes	Yes	Yes
	PageLaser12							
	150 dpi graphics mode	TOSLAS15.PR (b)	Yes	Yes	Yes	Yes	Yes	No
	300 dpi graphics mode	TOSLAS30.PR (b)	Yes	Yes	Yes	Yes	Yes	No
Xerox(Diablo)	630API	DIAB630A.PR	Yes	Underline	Yes	Yes	No	No

(a) This driver will print multi-colored graphs. (b) This driver is available from the Ashton-Tate Torrance Software Support Center. (c) This driver requires the additional files RSET.BAT and ASSIGN.FNT. (d) This driver requires the additional file CAN_150.EXE or CAN_300.EXE for extended graphics. (e) This driver requires the additional file GQ_150.EXE or GQ_300.EXE for extended graphics.

Usage Tips

@pmt

In *Framework II Advanced Topics* on page 1-140, the formula used by the **@pmt** function is incorrect. The formula should read

```
princ * rate / (1 - 1 / (1 + rate)^per)
```

Counting the Minutes

Framework II provides the **@time** function to record or set the time of day on a date item. However, no support is provided for arithmetic with times.

Addtime (see page 58) is a user-defined function that calculates the total of a list of times, stored in a range of spreadsheet or database cells. The times are lengths of times, as opposed to times of day. They are, therefore, represented as text instead of as part of a date item. Times are entered in the format

```
hours:minutes:seconds
```

Minutes and seconds are required, but hours are optional. An optional plus or minus sign may be placed in front of the time. Here are some examples of valid times:

1:30	One minute, thirty seconds
30:90	Thirty minutes, ninety seconds
14:24:6	Fourteen hours, twenty-four minutes, and six seconds
-7:42	Negative seven minutes, 42 seconds

Addtime should be on the desktop or in your Library when it is used. The syntax is

```
@Addtime(<range>)
```

where <range> is a range of spreadsheet or database cells. For example:

```
@Addtime(Week1.B2:Week1.B52)
@Addtime(Videos.Running Length)
```

@Addtime returns a character string that is the total of the items in the range. If no range is specified, #REF! is returned. If there is an error, such as a non-numeric character, in any of the times in the range, #VALUE! is returned.

Filtering Duplicates

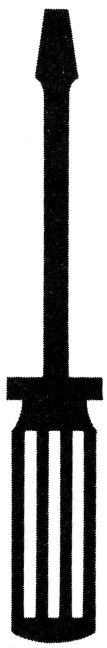
{Alt-D} is a macro that makes it possible to filter duplicate records from a Framework II database (see page 60). The macro assumes that you have highlighted a field in the database, and that the database has been sorted on the field to be checked for duplicates.

The macro adds a field named Tag to the database, and places an "X" in the Tag field of each unique record. It puts a filter formula in the border of the database frame, Tag = "X", which selects only the records that have been tagged. Finally, the Tag field is removed.

The filtered records can be returned by highlighting the database frame and toggling **Frames:Open All**.

Addtime.FW2

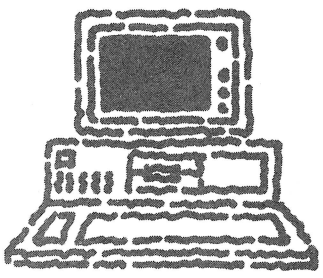
```
; Addtime.FW2
;   by Chuck Litzell
;   January 1, 1988
;
;   Calculates the sum of times in a range.
;   The times are entered as strings:
;
;             "hh:mm:ss"
;
;    The result is a string in the same format. Hours
;    are optional. Non-alpha's within the range are
;    ignored.
;
@local(hours, minutes, seconds, current, work, p1, sign),
;
@if(@itemcount < 1,
     @return(#REF!)
),
hours := minutes := seconds := 0,
;
; Walk through the range.
;
@while((current := @get(@item1)) <> #NULL!,
    ;
     ; If the cell is a string, try to add it in.
     ;
     @if(@isalpha(current),
          @list(
               ;
               ; Strip off leading spaces and plus sign.
               ;
               @while(@and(@len(current) > 1,
                           @scan(@mid(current, 1, 1), " +") <> #N/A!
                    ),
                    current := @mid(current, 2, 99)
               ),
               ;
               ; If the time starts with '-', remember it's
               ; negative for later.
               ;
               sign := @if(@mid(current & " ", 1, 1) = "-",
                           @list(
                                current := @mid(current & " ", 2, 99),
                                -1
                           ),
                           1
                    ),
               ;
               ; Not a valid time If there's not at least one colon
               ;
               @if((p1 := @scan(":", current)) = #N/A!,
                    @return(#VALUE!)
               ),
               ;
```



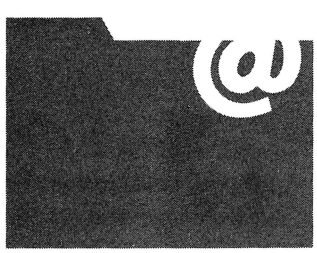
```
            ; Convert first part of time to a number in work.
            ; Then remove it from the time string.
            ;
            @if(@iserr(work := @value(@mid(current, 1, p1 - 1))),
                @return(#VALUE!)
            ),
            current := @mid(current, p1 + 1, 99),
            ;
            ; If there's another colon, it's a three-part time.
            ; We'll add the part we already have to hours, then
            ; get the next part in work.
            ;
            @if((p1 := @scan(":", current)) <> #N/A!,
                @list(
                    hours := hours + sign * work,
                    @if(@iserr(work := @value(@mid(current, 1, p1 - 1))),
                        @return(#VALUE!)
                    ),
                    current := @mid(current, p1 + 1, 99)
                )
            ),
            ;
            ; Add the minutes.
            ; Remainder of time string is seconds.
            minutes := minutes + sign * work,
            @if(@iserr(work := @value(current)),
                @return(#VALUE!)
            ),
            seconds := seconds + sign * work
        )
    ),
    @next(@item1)
),
;
; Adjust for overflows (> 60) in seconds and minutes.
;
minutes := minutes + @int(seconds / 60),
seconds := @mod(seconds, 60),
hours   := hours + @int(minutes / 60),
minutes := @mod(minutes, 60),
;
; Construct the result string.
;
@return(
    @if(hours * 3600 + minutes * 60 + seconds < 0, "-", "") &
    @if(hours <> 0, @integer(hours) & ":", "") &
    @if(@abs(minutes) < 10, "0", "") & @integer(@abs(minutes)) & ":" &
    @if(@abs(seconds) < 10, "0", "") & @integer(@abs(seconds))
)
```

{Alt-D}

```
; {Alt-D}
;
; By Quinn Wildman
;   January 1, 1988
;
; Filters a database frame to hide duplicate records.
; Before executing the macro, sort the database on the field
; you want to check for duplicates. Place the highlight anywhere
; in the field and then press {Alt-D}.
;
; The macro creates a new field, Tag, then marks duplicate records
; by placing an "X" in the Tag field. Finally, the filter formula,
; Tag = "X" is executed to filter the database.
;
; The filter formula and the Tag field are deleted before the
; macro finishes.
;
;
@local(current),
current := "",
;
;
; Make sure a database field is selected.
@eraseprompt,
@if(@frametype <> 12,
     @list(
          @beep,
          @prompt("Select a database field before executing this macro."),
          @return(0)
     ),
     @prompt("Filtering duplicates ... please wait")
),
@echo(#OFF),
@while(@frameview <> 1,
     @performkeys("{F10}")
```



```
),
; Create the Tag field, and "hide" it.
@performkeys("{Ctrl-PgUp}{Ctrl-C}C1{Return}Tag" &
                  "{Return}{F4}{Home}{Leftarrow}"),
@performkeys("{Return}{Leftarrow}{Dnarrow}"),
;
; Walk through the database, using Dnarrow
;
@isabend(#FALSE),
@while(@not(@isabend),
     ;
     ; if record is different than previous, mark it
     ;
     @if(@textselection <> current,
          @list(
               @performkeys("{Rightarrow}X{Backtab}"),
               current := @textselection
          )
     ),
     ;
     ; go to next record
     ;
     @performkeys("{Dnarrow}")
),
;
; Enter the filter formula.
@performkeys("{Uplevel}{F2}{F6}{Ctrl-End}{Del}Tag = ""X""{Return}"),
;
; Delete the Tag field.
;
current := @setselection,
@setselection(current & ".Tag"),
@performkeys("{Ctrl-E}CY{Return}{Leftarrow}"),
@echo(#ON),
@eraseprompt
```



Usage Tips

A Lookup Table

BY RICK FILLMAN

Developing a lookup table for a database system ordinarily means using a separate data file to hold the lookup table information. Since RapidFile allows the use of only one data file at a time, that approach isn't feasible. A lookup table can be simulated, however, by using calculated fields.

The *Using RapidFile* manual indicates how this might be done under the topic of "Lookup Calculated Fields" on page 13-16. The example given provides a type of data-validation scheme where if anything other than two specified strings is entered, the word "Invalid" is displayed.

This example can be expanded a bit further. Suppose a two-item lookup field is not sufficient. While more items could be handled in a single lookup field, this would require nesting the IFSTR$'s as far as needed. Keep in mind that RapidFile's field expression length limit is 254 characters. The solution lies in RapidFile's ability to have calculated fields that reference other calculated fields.

The following example uses multiple calculated fields. The example simulates a lookup table with approximately 60 items. The table allows user entry of a two-letter postal code for any of the 50 states, Washington D.C., or the provinces of Canada. The result displays the name of the state or province spelled out.

In the example, a two-letter postal code is entered into a field called "e". This is a simple text field. (The short field name was chosen so that we could cram as much as possible into subsequent expressions.) This entry field is referenced in turn by calculated fields that have been divided into two groups. The fields in the first group have been named f01, f02, f03, f04, and f05. The fields in the second group have been named f11, f12, f13, and f14. Figure 1 contain the expressions for each of these fields. The formulas have been formatted in the table to make the nesting clear, but when you enter them in RapidFile they should be entered as a single, continuous string with no returns or spaces, except where required in the state names.

Notice that this is no different than the example in the RapidFile documentation, except that the IFSTR$'s are more deeply nested. With states (taken alphabetically), there is room for eight items in this field. All the "f" fields reference the entry field in the same way. Only the content of the lookup items changes. Notice that if all the possibilities in these fields are exhausted, the fields return the single letter "x".

Next, a couple of calculated fields "gather" or consolidate from the results of the two groups. The field called "r0" gathers the results from "f01" through "f05". The field called "r1" gathers the results from "f11" through "f14". The formula in the first "gather" field, r0, says that if we are looking at a group of fields that are "x" in all cases (meaning that the lookup item was not found), then the result will be "x". On the other hand, if even one of the group was not an "x", then a lookup item was found. The "r0" field returns the name of the state or province.

Finally, the field called "state" references all the "gather" fields. In this example, only two "gather" fields need to be considered.

Notice that if both the "gather" fields are "x", then the word "Invalid" is returned. Otherwise, the lookup item from the gather fields is returned.

Once again, the principle we are applying is that calculated fields may reference one another. Of course, the "finished product" would never show any of these intermediate calculated fields. Only the entry field ("e" in our example) and the final result field ("state") would ever show on the screen.

You might notice that final closing parentheses have been omitted from the above formulas. RapidFile does not require closing parentheses in nested formulas.

Figure 1

```
Field: e        (Text)

Field: f01      (Calc, Text)

                IFSTR$([e="AL"],"Alabama",
                   IFSTR$([e="AK"],"Alaska",
                      IFSTR$([e="AZ"],"Arizona",
                         IFSTR$([e="AR"],"Arkansas",
                            IFSTR$([e="CA"],"California",
                               IFSTR$([e="CO"],"Colorado",
                                  IFSTR$([e="CT"],"Connecticut",
                                     IFSTR$([e="DE"],"Delaware",
                                     "x"

Field: f02      (Calc, Text)

                IFSTR$([e="DC"],"Washington D.C.",
                   IFSTR$([e="FL"],"Florida",
                      IFSTR$([e="GA"],"Georgia",
                         IFSTR$([e="HI"],"Hawaii",
                            IFSTR$([e="ID"],"Idaho",
                               IFSTR$([e="IL"],"Illinois",
                                  IFSTR$([e="IN"],"Indiana",
                                     IFSTR$([e="IA"],"Iowa",
                                        "x"
```

Figure 1 (cont)

Field: f03 **(Calc, Text)**

```
IFSTR$([e="KS"],"Kansas",
   IFSTR$([e="KY"],"Kentucky",
      IFSTR$([e="LA"],"Louisiana",
         IFSTR$([e="ME"],"Maine",
            IFSTR$([e="MD"],"Maryland",
               IFSTR$([e="MA"],"Massachusetts",
                  IFSTR$([e="MI"],"Michigan",
                     IFSTR$([e="MN"],"Minnesota",
                        "x"
```

Field: f04 **(Calc, Text)**

```
IFSTR$([e="MS"],"Mississippi",
   IFSTR$([e="MO"],"Missouri",
      IFSTR$([e="MT"],"Montana",
         IFSTR$([e="NE"],"Nebraska",
            IFSTR$([e="NV"],"Nevada",
               IFSTR$([e="NH"],"New Hampshire",
                  IFSTR$([e="NJ"],"New Jersey",
                     IFSTR$([e="NM"],"New Mexico",
                        "x"
```

Field: f05 **(Calc, Text)**

```
IFSTR$([e="NY"],"New York",
   IFSTR$([e="NC"],"North Carolina",
      IFSTR$([e="ND"],"North Dakota",
         IFSTR$([e="OH"],"Ohio",
            IFSTR$([e="OK"],"Oklahoma",
               IFSTR$([e="OR"],"Oregon",
                  IFSTR$([e="PA"],"Pennsylvania",
                     IFSTR$([e="RI"],"Rhode Island",
                        "x"
```

Field: r0 **(Calc, Text)**

```
IFSTR$([f01="x"]
      .AND.[f02="x"]
      .AND.[f03="x"]
      .AND.[f04="x"]
      .AND.[f05="x"],
   "x",
   IFSTR$([f01<>"x"],f01,
      IFSTR$([f02<>"x"],f02,
         IFSTR$([f03<>"x"],f03,
            IFSTR$([f04<>"x"],f04,
               f05
```

Figure 1 (cont)

Field: f11 (Calc, Text)

```
IFSTR$([e="SC"],"South Carolina",
   IFSTR$([e="SD"],"South Dakota",
      IFSTR$([e="TN"],"Tennessee",
         IFSTR$([e="TX"],"Texas",
            IFSTR$([e="UT"],"Utah",
               IFSTR$([e="VT"],"Vermont",
                  IFSTR$([e="VA"],"Virginia",
                     IFSTR$([e="WA"],"Washington",
                        "x"
```

Field: f12 (Calc, Text)

```
IFSTR$([e="WV"],"West Virginia",
   IFSTR$([e="WI"],"Wisconsin",
      IFSTR$([e="WY"],"Wyoming",
         "x"
```

Field: f13 (Calc, Text)

```
IFSTR$([e="AB"],"Alberta",
   IFSTR$([e="BC"],"British Columbia",
      IFSTR$([e="MB"],"Manitoba",
         IFSTR$([e="NF"],"Newfoundland",
            "x"
```

Field: f14 (Calc, Text)

```
IFSTR$([e="NS"],"Nova Scotia",
   IFSTR$([e="ON"],"Ontario",
      IFSTR$([e="PE"],"Prince Edward Island",
         IFSTR$([e="PQ"],"Quebec",
            IFSTR$([e="SK"],"Saskatchewan",
               "x"
```

Field: r1 (Calc, Text)

```
IFSTR$([f11="x"].AND.[f12="x"].AND.[f13="x"].AND.[f14="x"], "x",
   IFSTR$([f11<>"x"],f11,
      IFSTR$([f12<>"x"],f12,
         IFSTR$([f13<>"x"],f13,
            f14
```

Field: state (Calc, Text)

```
IFSTR$([r0="x"].AND.[r1="x"],"Invalid",
   IFSTR$([r0<>"x"],r0,
      r1
```

A Minimum Date Function

RapidFile has no minimum date function. The MIN function is a numeric function and works on numeric values only. This means that if you want to locate the oldest date in a group of records, you must write your own formula.

To do this, start with a calculated numeric field that will split the date value into its individual components – year, month, and day. Each of these is converted from numeric to string data, and the IF$ function is used to pad a leading "0" where the resulting string is a single digit. Then the three strings (YEAR, now a four-character string, MONTHS, now a two-character string, and DAY, also a two-character string) are concatenated. The result is made numeric again by the NUM function.

The numeric value created by this formula will look like "19871030".

Next, use the MIN function in a report field to select the minimum value from the group of records associated with a particular value in the grouped field.

Since this is a report field, the MIN values will not appear until an appropriate sort and recalc are done

(RECORDS:Order:Implement and RECORDS:Calc report fields).

Finally, the MID$ function in combination with STR$ is applied to the result of the MIN function, along with slashes to get the desired format for the final result. The result displayed, as a calculated text field, is the value that represents the oldest date.

Figures 2 and 3 illustrate the fields and expressions needed to implement this strategy.

Note: RapidFile will not automatically display the calculated result in this field. Since this calculated field is referencing a report field, you must do a Ctrl-I (Instant Recalc) to see the values.

Field name	Type
Lastname	Text
Date	Date
calc_num	Calc,N0
oldest_num	Rprt.0
calc_date	Calc,T

Figure 2

calc_num Expression Formula

```
NUM(STR$(YEAR(date))+
IFSTR$(LEN(STR$(MONTH(date)))=1,"0"+STR$(MONTH(date)),STR$(MONTH(date)))+
IFSTR$(LEN(STR$( DAY(date)))=1,"0"+STR$( DAY(date)),STR$(
DAY(date)))
```

oldest_num Expression Formula

```
MIN:calc_num(Lastname)
```

calc_date Expression Formula

```
MID$(STR$(oldest_num),5,2)+"/"+
MID$(STR$(oldest_num),7,2)+"/"+
MID$(STR$(oldest_num),3,2)
```

Figure 3

Anomalies

dBASE III PLUS

Applications Generator

Including a database file with the field name "Item" hangs the system when option 6, Automatic Applications Generator, is run from the APPLICATIONS GENERATOR main menu. When option 6 is run, answering any prompt except "Enter APPLICATION AUTHOR:", "Enter APPLICATION filename:", and "Enter DATABASE filename:" hangs the system before the application code is generated. To work around, rename the offending field. This occurs in all versions of dBASE III PLUS.

CREATE/MODIFY STRUCTURE

It is possible to define an illegal field name during CREATE/MODIFY STRUCTURE. To reproduce this, CREATE a database file and create a field name called My_number. Use **Leftarrow** to position the cursor under the letter "M" of My_number and press **Del** twice to leave the field name _number, then press **Ctrl-W** to save the file. dBASE III PLUS accepts the invalid field name. This file can be used for all full-screen editing commands, but any commands that explicitly reference the invalid field name return the error message "Syntax error." [10]. This occurs in all versions of dBASE III PLUS.

dCONVERT

When dCONVERT version 2.01 converts an index file located on another drive, the new file has an .NDX extension. This is actually a mislabeled .RX file. To convert a file on another drive, use the command

```
A>DCONVERT B:<Index file>.NDX
A>RENAME B:<Index file>.NDX B:<Index file>.RX
```

dCONVERT converts an index file to a command file with an .RX extension. This is because dCONVERT cannot actually create an .NDX file. The .RX file contains an INDEX ON command and is executed by first USEing the correct database and then entering the command DO <Index file>.RX

---

dCONVERT does not always recognize "rank" as a memory variable name. It may translate this word to "ASC" without parentheses, when converting to dBASE III. For example,

```
STORE 1 TO rank
? rank
```

will be dCONVERTed to

```
STORE 1 TO rank
? ASC
```

---

dCONVERT does not always recognize "type" as a memory variable name. It may interpret this word as a function when converting to dBASE III PLUS. For example,

```
STORE 1 TO type
? type
```

is dCONVERTed to

```
STORE 1 TO type
? type()
```

EColor43

In the dBASE III PLUS Version 1.1 3.5" version on the IBM PS/2 Models 50, 60, or 80, the RUN/! command writes to lines higher than line 37 when SET Screen:Display type is set to EColor43, enhanced 43-line mode. However, CLEAR does not clear lines greater than line 37. RUN CLS does not clear these lines either. This occurs only on the 3.5" version of dBASE III PLUS

Memo Fields

dBASE III PLUS appears to allow data to be entered into a memo field in a work area other than the currently selected work area. However, the data is not saved. For example, assuming that the current work area is 2, READing a format file with the line

```
@21,3 GET A->Memofield
```

appears to save the edited record. After editing the memo field contents the message "File being saved." briefly flashes

on the screen. However, displaying the memo field reveals that the data was not saved. This is corrected in version 1.1 of dBASE III PLUS.

READKEY()

The original *Learning and Using dBASE III PLUS* manual states on page U6-47 that READKEY() returns values of 9 and 8 for **Ctrl-Rightarrow** and **Ctrl-Leftarrow** respectively. However, these keys cannot be used to exit a full-screen mode and therefore cannot be trapped in this manner.

Runtime — CONFIG.RT

Analogous to the dBASE III PLUS CONFIG.DB file is the dBRUN CONFIG.RT file. dBRUN III PLUS processes CONFIG.RT upon execution to determine its initial configuration.

When HISTORY or DOHISTORY is included in the CONFIG.RT file, dBRUN III PLUS returns the error message "Keyword not found. Press any key to continue...." After a key is pressed, dBRUN III PLUS continues to execute.

Runtime — dBCODE

When ENDTEXT is the last command in a program file and the command line is not terminated with a **Return**, the error message "Premature end of input file <Filename> - input line <n>." is produced when the file is dBCODEd.

When TEXT is the last command in the program file and ENDTEXT is missing from the TEXT...ENDTEXT structure, the error message repeats continuously, requiring a warm boot. For example,

```
* Test1.SRC
TEXT
   Some strange brew.
ENDTEXT
       ^--There is no carriage return/linefeed here.
* Test2.SRC
TEXT
   I think I'm going to Kathmandu.
* EOP: Test2.SRC
```

When Test1.SRC is dBCODEd, the error message is displayed and dBCODE terminates normally without producing a .PRG file. However, dBCODEing Test2.SRC requires a warm boot when the error message is displayed. This occurs in version 1.1 only.

Runtime — RETRY

RETRY causes problems if it is dBCODEd and executed. If a RETRY command is encoded with dBCODE, the machine locks when the command is executed, requiring a cold boot. The following programs demonstrate this when encoded and run.

```
* Retro.SRC
DO WHILE .T.
  ? 'THE FIRST FILE'
  DO B:Retro1
ENDDO

* Retro1.SRC
ON ERROR RETRY
? 'THE SECOND FILE'
* No variable by this name exists.
? asfasf
RETURN
```

If SET ECHO ON is added to Retro.SRC, the computer does not always lock, but the DO WHILE loop terminates after one iteration, even though there is no EXIT or RETURN command.

FRAMEWORK II

Alt-224 Key

The new IBM keyboard driver does not generate a character for **Alt-224** when using the numeric keypad. To generate this character, you will need to create a macro. For example,

```
; This generates the IBM character for Alt-224
@performkeys("{224}")
```

This occurs in all versions of Framework II except the 3.5" format of version 1.1.

Alt-F8 Key

Framework II hangs if the **Alt-F8** Paste macro is executed while editing a formula in cursor–pointing mode. This occurs in all versions of Framework II.

Exporting to WordStar

Framework II does not properly export attributed text (bold, underline, and italics) if the text includes a carriage return and the next line does not use the same attributes. The export program processes a line only up to the carriage return. If the carriage return is attributed, the Framework II sequence to turn the attributes off occurs after the carriage return; therefore, the export program never processes the sequence that turns the attributes off. For example, a line that is boldfaced up to, but not including, the carriage return exports properly. If the carriage return is boldfaced, however, the bold attribute is not turned off at the end of the line. The result is that the remaining text exports in boldface unless bold is set ON and then OFF somewhere else in the text.

To work around this problem, turn **Edit:Display Hidden Characters** ON, highlight any carriage return that is attributed and turn **Words:Normal** ON before exporting.

This occurs in all versions of Framework II.

Exporting to WordStar

Framework II does not export centered text or inserted page breaks to WordStar ®. Centered text is aligned left and inserted page breaks are ignored.

To export an inserted page break to WordStar, delete the inserted page break in Framework II, replace it with the WordStar .PA dot command, and then export the frame. The .PA is exported as text and the WordStar program interprets it as a page break. There is no way to export centered text as centered text. This occurs in all versions of Framework II.

Outline View

While in outline view, pressing EXTEND SELECT (F6), **Dnlevel**, and **Del** from a parent frame causes the system to freeze requiring a cold boot. To demonstrate this, create an outline frame containing four sections, each of which contains three subsections (see Figure 1).

As this shows, the problem occurs only when either (1) section 4 or beyond is highlighted, or (2) the highlighted subsection contains as many child frames as the last digit of its outline number. Using **EXTEND SELECT (F6)** to select the child frames is not necessary. Deleting the parent frame deletes its child frames. This occurs in all versions of Framework II.

View Pagination

Moving text, with **MOVE (F7)**, within a word frame when **Frames:View Pagination** is ON can corrupt the frame under the following circumstances. If the source text immediately follows a page break that is not an inserted page break and contains a carriage return, and the destination is the first character of the first page, the contents of the frame are replaced with a series of graphic characters when the move is performed. Paging through the frame using the **PgDn** and **PgUp** keys displays more corruption.

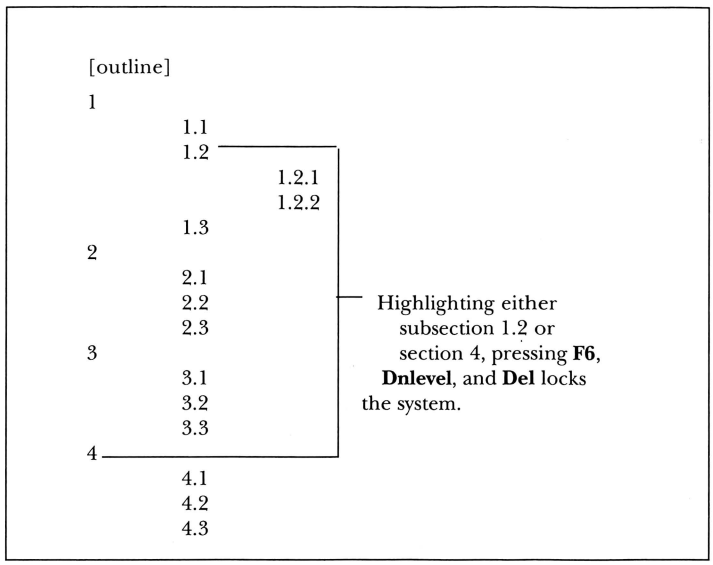


Figure 1

The text may still be retrieved if the following steps are executed before performing any other operations on the corrupted contents of the frame.

1. Press **Ctrl-Home**.

2. Using **Ctrl-PgDn**, move down as far as allowed.

3. Using **Ctrl-PgUp**, move up until the corruption reverts to text.

4. Continue to use the **Ctrl-PgUp** until the top of the file is reached.

5. Use **EXTEND SELECT (F6)** and **Ctrl-End** to highlight all the text, and then use the **Alt-F7** Cut macro to cut to the Cut Paste buffer.

6. Create a new frame and use the **Alt-F8** Paste macro to paste the text into the new frame.

Note that if any of the corrupted text is copied, moved, or deleted before the above steps are performed, the results may vary.

This occurs in all versions of Framework II.

Custom Reports

Page 12-10 of *Using RapidFile* states that group headers are printed after the header at the top of every page and after each sub-section. In fact, group headers appear only after each sub-section.

Form Layouts

The **Print:Print settings:Wait before each page** option does not work when printing multi-page form layouts. The prompt to insert paper is returned before the first page of the form is printed but is not returned for subsequent pages.

To work around this problem, first print the form to a DOS text file. Then retrieve the text file in the Memo Writer and print it with the **Print:Print settings:Wait before each page** option set ON.

Form Letters

Under **Creating Form Letters** on page 18-2 of *Using RapidFile*, you are instructed to place the cursor where the merged data is to begin and then choose the **Merge:Insert field** option to place the field in the letter. However, if the cursor is moved past the end-of-line marker with the **Right-arrow** key and a field is inserted, the new field will be positioned starting at the end-of-line marker. You may use the spacebar to move to the position where you want the new field.

Operators

Pages 10-11 and 10-12 of *Using RapidFile* illustrate the use of comparison operators in Tables 10-2 and 10-3. Although the examples appear to indicate a space between the operators when more than one operator is used, spaces should not be included. RapidFile selects all records when spaces are used between the operators in the match criteria.

Quietwriter

When using RapidFile's Memo Writer with the IBM Quietwriter printer driver IBMQUIET.PR installed, the lines per inch setting in the printer is set to 4.5 instead of 6. This changes the line spacing to approximately 1.5 lines per inch, which throws off the page length. After returning to the File Manager, all subsequent printed output is at 1.5 lines per inch. The printer must be turned off to reset the line spacing.

The problem occurs because the Variable Line Feed code in the printer driver is incorrectly set to 10 hexadecimal.

Putting Printer Codes in the Memo Writer does not correct the problem because the Variable Line Feed code is the last one sent before the text.

To use the Memo Writer with the IBM Quietwriter, install the RFQUIET.PR printer driver which has the correct Variable Line Feed code. This driver is available from the Ashton-Tate Torrance Software Support Center.

To install the new printer driver, use the following procedure.

1. Go to DOS and make \RAPID the current directory (CHDIR \RAPID).

2. Put the diskette with the new printer driver in the A: drive and type

```
COPY A:RFQUIET.PR C:\RAPID\PRINTER1.DRV
```

Corrigendum

Converting to Initial Caps

December 1987, page 40

There is a typographical error in the line immediately preceding the "DO CASE". This line should read:

```
pchar = SUBSTR(tempstr, counter, 1)
```

On-Line Services

By Steve Silverwood

Do you have a modem? Do you subscribe to CompuServe or The Source? If so, Ashton-Tate offers technical support on all of these electronic information services. You can quickly access the kind of quality technical information that you have come to expect from TechNotes, and you can do it through your computer system. To take advantage of the progressive support method, you must subscribe to one of the services listed below. There are no additional charges for using our on-line support services, apart from the normal connect-time fees for each service. You can get specific information by calling one of the following numbers:

Subscriber Service Information		**Command to find Ashton-Tate**
CompuServe	(800) 848-8199	GO ASHTON
The Source	(800) 336-3366	ATSIG

To give you an idea of the support that is happening on-line, the following are regular excerpts from the Ashton-Tate Forum on CompuServe. This monthly material comes from messages our Forum users have posted on the bulletin board. All message headers have been removed to protect the identities of our users. Every other message within a single conversation topic has been italicized to enhance readability.

dBASE III PLUS

132-Column Display

I am using an AST Preview I/O board, which is capable of displaying 132 columns by 44 rows. It works fine with Lotus 1-2-3 and WordPerfect. Is it possible to use dBASE III PLUS with this card so more information can be displayed with BROWSE?

dBASE III PLUS does not support 132 columns. It works in 43-line mode (as you probably found out), but it has no capability for more than 80 columns.

Testing for Files

I want to write a program that will store the times and dates of use for each user on the system. The real problem happens when a new user enters his initials. For example, if the prompt "enter initials" is displayed, the person enters "DFG". The system should then check to see if there is a file named DFG.DBF If so, fine. If not, instead of giving an error message the system should create a file by that name. I know how to create a file automatically, but I don't know how to intercept the error messages without a specific one appearing on the screen.

Try using the FILE() function to see if the file exists. If it doesn't, then use the CREATE ... FROM command to create it from a standard file format:

```
IF .NOT. FILE("&Initials..DBF")
  CREATE &Initials FROM Standard
ENDIF
```

Keeping Record Numbers Sequential

I have a dBASE file with the following fields: LastName, First-Name, City, and CardNo I have indexed on the LastName field. Every time I add a new name, it is listed in alphabetical order, which is what I want. But, what I would like to do is have the CardNo field continue numerically in order along with the LastName being alphabetical. I would like this to happen even when I add new names or delete old ones.

I have a database like this:

LastName	CardNo
Andrews	1
Bowers	2
Castle	3
Davis	4

If I add another name that would fall in that group alphabetically, it would start the #1 CardNo at the first alphabetical name like this:

```
LastName            CardNo

  Andrews             1

  Bowers              2

  Castle              3

  Connors             4   (New one!)

  Davis               5
```

It would automatically insert the new name and renumber the CardNo field. Can you suggest how I would do this either in the database or in a program file?

Assuming the record pointer is still on the new record...

```
mrecord = RECNO()
GO BOTTOM
mtotal = RECCOUNT()
DO WHILE RECNO() <> mrecord
   REPLACE CardNo WITH mtotal
   mtotal = mtotal - 1
   SKIP -1
ENDDO
REPLACE CardNo WITH mtotal
```

Delayed Messages

I have written a dBASE III PLUS application that displays messages on the screen for 2 or 3 seconds. I use a DO WHILE loop and count to 70 or 100 to get the right timing for the PC/XTs that the application is running on. Of course, when the application runs on an AT (or anything else with a faster clock speed) the messages are barely readable before they vanish. Is there a way to use the system time from within dBASE III PLUS to time the messages so they will appear for the desired time regardless of the clock speed of the computer? I don't want to use WAIT or any other interactive means of displaying the message long enough to be read. These messages are confirmations that an action has taken place or mild warnings to watch out for something. Not the kind of thing where the user has to ponder his course of action.

One approach is to use an environment variable that's set differently for each machine. It can be a character that will relate to a performance factor or it can just be a number. For instance, SET DELAY=100 for XTs and SET DELAY=300 for ATs. You can read this with dBASE's GETENV() function. Place the SET command in AUTOEXEC.BAT file (or the batch file that calls your program). Your program would have a line like:

```
mdelay = VAL(GETENV("DELAY"))
```

"delay" is then your dBASE memory variable for the counter. If that seems too awkward here's another timing routine:

```
PARAMETERS seconds
elapsed = -1
DO WHILE seconds > elapsed
   elapsed = elapsed + 1
   temp = TIME()
   DO WHILE temp = TIME()
   ENDDO
ENDDO
RETURN
```

TYPE()

Have I found another undocumented feature or am I doing something wrong here? Try this:

```
mnum = SPACE(5)
@ 5,5 GET mnum PICTURE "xxxxx"
mnum = TRIM(mnum)
? TYPE(mnum)
```

Mnum is almost always a number, such as '123', but it hasn't been declared a numeric variable. ? TYPE(mnum) returns an "N." Why? If I DISPLAY MEMORY, mnum is listed as a character variable. This is causing inconsistent data-type mismatches.

The TYPE() function attempts to reduce its argument, which is a character expression, to a dBASE data type. The result is a one–character code for the data type: "D" for Date, "N" for Number, "L" for Logical, "M" for Memo, or "U" for Undefined (or Unknown). The argument can be an arbitrarily complex dBASE expression.

In your program, if the user enters 123 into mnum, TYPE(mnum) interprets the digits in mnum as the numeric value 123, and the result of TYPE() is "N". If the user enters 123A into mnum, the result if TYPE() would be "U" since 123A doesn't fit any of the dBASE data types. Another way to look at this puzzle is to draw the parallel between TYPE() and macro expansion. TYPE() "expands" and "evaluates" its argument much as the dBASE interpreter does when a memory variable is prefixed with the ampersand. If you typed

```
mnum = "123"
? &mnum
```

at the dot prompt, the numeric value 123 would be displayed, since the macro operator transforms the command to

```
? 123
```

before it is evaluated. Similarly, if you type

```
mnum = "123"
? TYPE(mnum)
```

an "N" will be displayed, because TYPE() sees the numeric value 123 in its argument. If you just want to check the data type of a memory variable, surround the memory variable with quotes

```
mnum = 123
? TYPE("mnum")      && Displays "N"
mnum = "123"
? TYPE("mnum")      && Displays "C"
```

FRAMEWORK II

Color

I'm having trouble getting Framework II to run in color. I have a Panasonic FX-BG171 color card and a Sears color monitor attached. This system works well with other software. In **Setup** I selected the IBM Color Graphics (Text) driver. Background and foreground are set to their defaults (0). Only the command line displays in color. Everything else, including the sign-on screen is white on black.

The default values for foreground and background colors are a white display on a black background. I believe the logo will stay in black and white, but the other colors can be changed by changing the default color numbers in Setup. There's a file in DL3 that gives numbers for different color combinations.

Using the standard IBM Color Graphics screen driver in Framework II, the background must be Black (0), and you can set the foreground to one of eight colors. However, if you use the text mode driver, you can select colors for both foreground and background.

DOS Access

I am learning Framework II and have a question on DOS access. I use Framework II on a desktop IBM-XT compatible running MS-DOS 3.2. When I call for DOS access, Framework II opens a DOS access frame, but then the screen clears and I get the DOS prompt. On issuing EXIT from the DOS prompt my Framework screen reappears with no text captured in the DOS access frame.

On an AT compatible laptop, running the same version of MS-DOS, the DOS access remains within the frame and the text is captured. Why the difference?

This will happen if you have any resident program that clears the screen when a new DOS command shell is loaded. ANSI.SYS will do this, as will many DOS enhancers, like PCED, DOSEDIT, and NE.

This is the expected behavior for Framework II. You should check your AUTOEXEC.BAT and CONFIG.SYS files for offenders and remove them if this is important to you.

It is also worth noting that if you are running Framework II while connected to certain LANs, accessing DOS may also result in the "screen clearing" action. I assume this happens because the software running on the LAN clears the screen. There is no way around this that I know of.

Abbreviations/Zenith

I am having trouble using abbreviations in Framework II (version 1.1). When I invoke the abbreviation by typing **Alt-Backspace**, the abbreviation is typed in, but the last word of the abbreviation is erased. Also, if I invoke **Alt-Backspace** with no valid abbreviation, Framework II gives me the message that no abbreviation is on file and proceeds to erase the word to the left of the cursor. Is this a bug or am I doing something wrong? I have worked around the first part by putting an extra word at the end of each abbreviation. This seems to happen with the sample abbreviations provided in the Library as well. I am running Zenith MS-DOS 3.2 on a ZENITH 151 .

*This reflects a small incompatibility in the Zenith BIOS. I have always told people to include an extra "word" in their macros (just an "x" will do, for example). Another answer is to add a macro to your Library Cabinet labeled {**Alt**-<keyname>}, where <keyname> can be any alpha/numeric keystroke you choose. In the formula area of the frame, type*

```
@performkeys("{Alt-Backspace}")
```

Instead of pressing {Alt-Backspace} *to expand an abbreviation, press* {Alt-<keyname>}. *This will actually perform an {Alt-Backspace}.*

43-Line EGA Screen Driver

How can I install the 43-line EGA screen driver available in the data library?

When you've downloaded the file, put it on a floppy disk and place it in drive A. Then go into your Framework II subdirectory and run the Setup program. From the first menu. select "Option 2: All other uses...".

*When you've told the program where to find the FWSETUP file, you'll reach the Main Menu. Select Option 2 for Configuration, Option 2 for Primary Hardware, then 1 for Screen driver. Next, enter 7 for, "I want to enter my own driver file name". The response to the next prompt is EGA43.SC the name of the file on the floppy disk. The system will return you to the menu. Type M to get to the Main Menu, then 7 to Save changes. It will prompt you to insert the Setup disk in drive A. Ignore this, as it's only looking for the driver that's already on the A drive. Press the **Spacebar** to have Setup load the driver.*

At the next menu, select the appropriate location for the FWSETUP file to be stored, then the program will return you to DOS. At this point, you can start Framework II and it will come up in 43-line mode.

*Note also that this is a text driver, so you won't be seeing any underlining, boldface, or italics, nor will there be any high-resolution graphics unless you **ZOOM (F9)** a graph to full screen. Instead, what you will see are color changes and intensity changes for text attributes, and character representations of graphs. If you're not doing much in the way of graphics, then you'll find the EGA43 driver to be very handy.*

Super/Subscripts

I have an Olympia Compact NP that supports superscripting and subscripting. Unfortunately, I can't seem to get my copy of Framework II to do the same. Any suggestions?

If there is a printer driver on the Setup disk for your printer, or one with the same control codes, you can get superscripts and subscripts in Framework II by entering special codes into the text.

To turn on superscripts, press ***Alt-244***

To turn off superscripts, press ***Alt-245***

To turn on subscripts, press ***Alt-246***

To turn off subscripts press ***Alt-247***

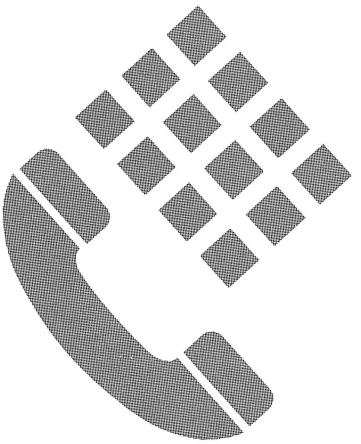
Alt-### *indicates that you hold down the* ***Alt*** *key and enter the numbers on the keypad. You could create some macros to do this more easily using the Fred* ***@chr()*** *function.*
If this has no effect when you print the text, the printer driver you're using doesn't know the control codes for your printer. You need to install another driver, or use the Framework II Developer's Toolkit *to add the necessary codes.*

RapidFile

Snow on AT&T 6300 Color

I just purchased RapidFile and think it is great! However, I am having a problem with video snow. I use an AT&T PC 6300 with a color monitor. I placed SET BLANKING=ON in my AUTOEXEC.BAT file as recommended in the READ.ME file on the distribution disk, but still get snow. This problem persists with nothing else in CONFIG.SYS or AUTOEXEC.BAT and no RAM-resident programs loaded. None of the other programs I use have this problem.

I'm afraid that a little snow with RapidFile is unavoidable on the 6300 (I have one). If you SET BLANKING=ON, then the snow problem is greatly reduced. On a color monitor, though, a little snow will always be visible.



Letters

dBASE III PLUS

New DIRECT/36

Dear *TechNotes*: I understand that Ashton-Tate has a software package that will let you interface dBASE III PLUS running on a microcomputer with an IBM SYSTEM/36 minicomputer. Is this true? What is the name of the product and what do you need to run it on a PC?--QD

Dear QD: Ashton-Tate recently announced dBASE DIRECT/36, a connectivity software package that links dBASE III PLUS with data residing on the IBM SYSTEM/36. To run DIRECT/36 on the PC, you need an IBM PC, PC/XT, PC/AT, or 100% compatible with a color or monochrome monitor, the IBM enhanced 5250 Emulation program version 2.1 or 100% compatible, PC DOS 2.xx through DOS 3.3, a copy of dBASE III PLUS, and at least 640K of RAM.

The IBM System/36 requires SSP release 5.0, PC Support/36, and the Interactive Data Definition Utility (IDDU).

Installing DBA with STARLAN

Dear *TechNotes*: We recently installed dBASE III PLUS (version 1.1) into an AT&T STARLAN Network (version 2.0). We are trying to set up dBASE so that it will be implemented as a turnkey system. We are trying to enter the parameters to load dBASE ADMINISTRATOR into the "Add Tasks" of the StarLAN Menus service. We set up a shared directory called dBASE on the server and linked a virtual drive (D:) to it. We specify D: both as the file location and as the startup run location in the "Add-Task" form. When prompted for the parameters we enter #DF=D:, but we have not been able to make it work correctly. Can you give us some advice as to how to set it up?--SU

Dear SU: To install the dBASE ADMINISTRATOR as a task in the StarLAN Menus Service, I recommend that you install DBA as a subdirectory off the root in the server. If all the files in that subdirectory are flagged as read-only (i.e. ATTRIB +R C:\DBA), if the subdirectory is shared in the Network (i.e. FS ASHARE DBA=C:\DBA), and if the subdirectory is linked to a virtual drive (i.e. FS LINK D: \\SERVER\DBA), then specifying the following settings entered at the "Add Tasks" prompt should give you successful results:

```
File Location: D:
Run File or Command: DBA
Run File Type: Program
Startup Run Location: D:
Pause after Run?     N
Keep menus program in memory while
running?    N
```

Do not *enter any parameter prompts or parameter values in the table in the lower part of the form. These are not necessary since a virtual drive has been linked to that directory for the session, making it the current directory.*

Missing .DBT File

Dear *TechNotes*: We recently copied a database file into our dBASE III PLUS subdirectory. When we go into dBASE and attempt to USE the file, we get the error message "DBT file cannot be opened," Can you tell us what is wrong and what we can do to access the data in our files?--MM

Dear MM: The database file you are trying to open contains one or more memo fields in its structure. The memo field data is stored in a .DBT file with the same name as the .DBF file (the database). Probably, when you copied the .DBF file, the .DBT was not copied. The solution is simply to copy the .DBT file into the subdirectory where its twin .DBF file is stored.

If your .DBT is missing or erased, create a dummy .DBF file with any name (one memo field and no data will suffice). Next, RENAME the new .DBT file (either from DOS or from dBASE) so that it will have the same name as your .DBF file, but extension .DBT. This technique will let you open the database file, although it won't let you access your previous memo field data.

Corrupted Index

Dear *TechNotes*: We are running a multiuser application on a 3COM 3PLUS Network (version 1.0.1) using dBASE III PLUS version 1.0. Every time we edit a database file that contains less than 100 records we experience several problems. On occasion, the LISTing of the records in the file will display only every other record. Sometimes it returns the error message "Illegal Opcode." Do you know why this happens, and do you have any suggestions as to how we can avoid this problem?--CI

Dear CI: Editing any field while using a database file with two index files open and SET EXCLUSIVE OFF can corrupt the first index file. If a modification is made to the key of the first index file, problems like the ones you describe will result.

A workaround to this problem is either to USE the file with the index open in exclusive mode (i.e. USE <Filename> INDEX <Index List> EXCLUSIVE) or to edit the database file with only one index file open. You should also consider upgrading to dBASE III PLUS version 1.1, especially on a network. Aside from the improved indexing, the removal of copy protection in version 1.1 greatly simplifies installation and access to the ADMINISTRATOR. The version 1.1 LanPack is installed on the server, so your individual users do not need to have their own local copy of Access, as they do with version 1.0.

Printer Buffers

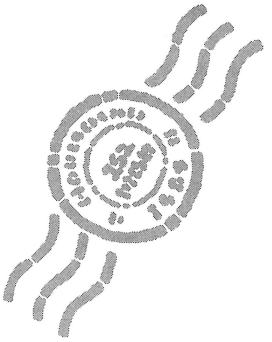
Dear *TechNotes*: I am running a multiuser application on a PC-LAN Network using dBASE III PLUS. When running a program that prints a report, the report is not printed until another print task is sent to the printer. Why does this happen and how can I prevent it?--PB

Dear PB: When a Network Spooler is active, output is kept in a buffer on the file server and is not immediately sent to the printer. To force output to the printer, execute a SET PRINTER TO <device name> command (i.e. SET PRINTER TO LPT1), this will cause PC-LAN to flush the spool buffer to the network printer.

Undocumented Error Message

Dear *TechNotes*: I am running a multiuser receivables system on the Novell Network (Novell Advanced Netware 1.01). When one of my programs is searching for a record in one of the indexed databases, dBASE III PLUS returns the message: "Relation record in use by others." I looked in my documentation, but I could find no mention of this message .--RL

Dear RL: "Relation record in use by others." is an undocumented dBASE ADMINISTRATOR error message (error message number 142). It's returned when an attempt is made to position the record pointer to a record that is not locked, but that has a relation set to a record that is locked by another user. Only GO and SEEK return the error. GO TOP, GO BOTTOM, and SKIP do not. When this error is displayed, the record pointer in the linked database file is not moved.



FRAMEWORK II

@sense

Dear TechNotes: Is there any way of knowing from within an application whether Telecommunications is active or not?--DT

Dear DT: The formula @sense(#ATSB) returns either the active telecommunications frame name, or #N/A! if there is no active telecommunications frame. An expression like:

```
@if(@len(@sense(#ATSB))=0,
     @do_active,
     @do_inactive
)
```

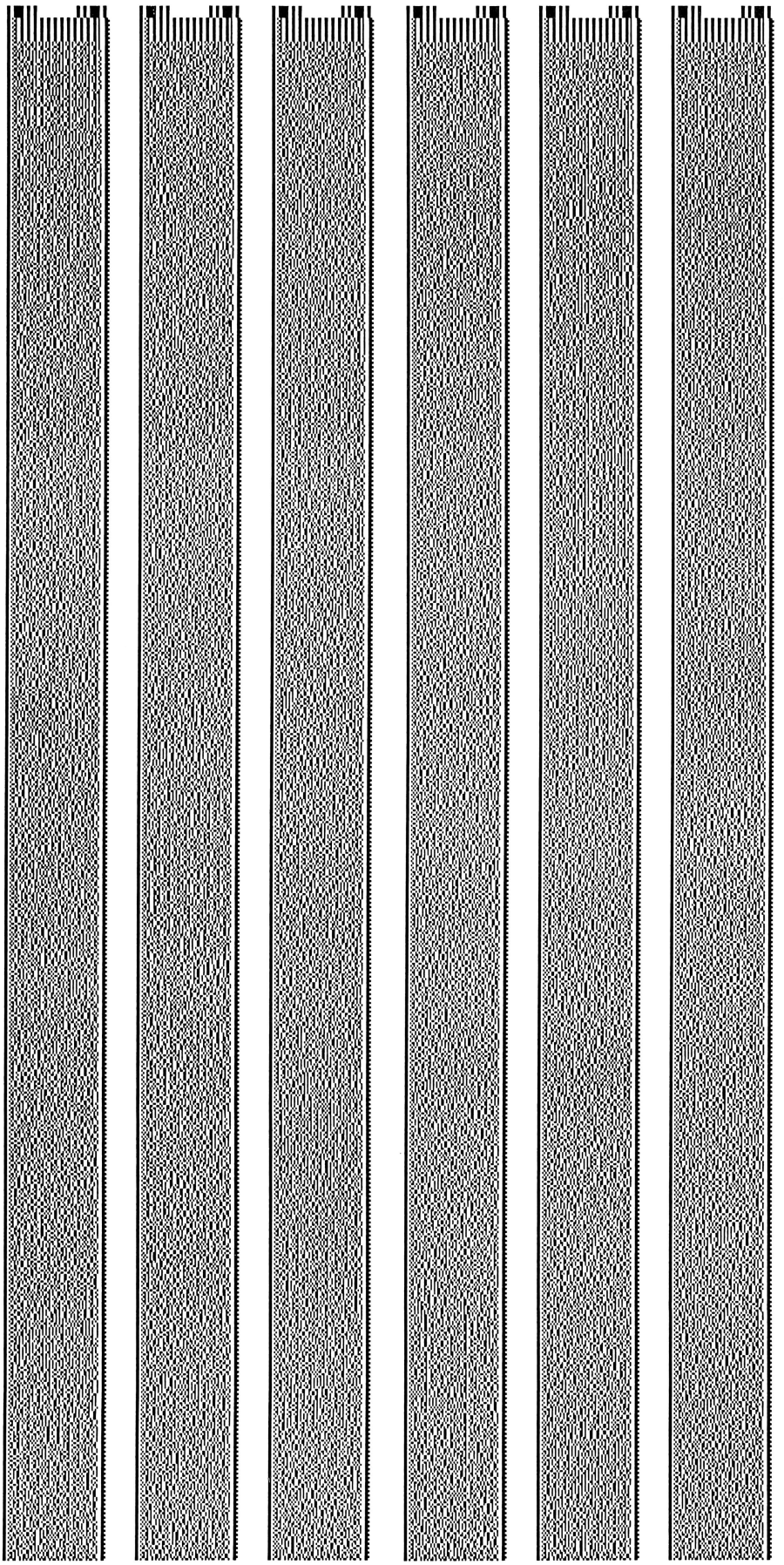
can be used from within a program to follow alternative courses of action in the program based on the telecommunications status.

Updating Formulas

Dear TechNotes: I am working on a spreadsheet created with Framework II version 1.1. When I remove some of my columns, the formulas on some of the remaining columns do not get updated and show a #REF! error. What's wrong?--FR

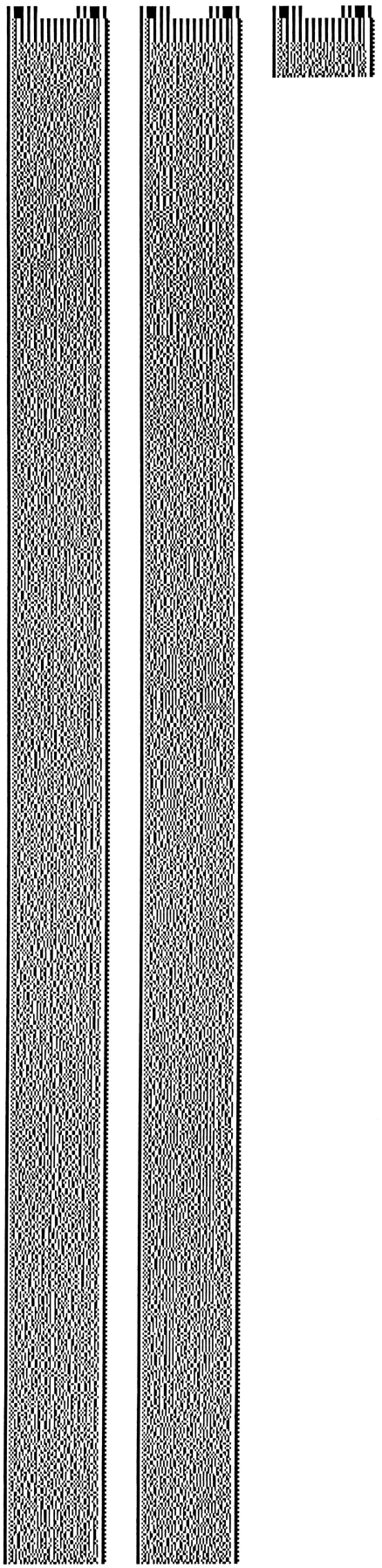
Dear FR: In Framework II a cell reference is not updated if it refers to a cell in a deleted row or column. Instead, the reference is replaced with the error constant #REF!(), with the original reference as the argument for the error constant. This is to call your attention to the potential error of a formula referring to a cell that has been removed. For example, suppose you have a spreadsheet with the formula A1:=10 in cell B1 and you copy it to cell D1. It will be updated to the formula C1:=10. If you remove column C, cell D1 becomes cell C1 and the original formula (C1:=10) becomes #REF!(C1):=10. The cell C1 that you referred to no longer exists, so Framework II is calling your attention to that problem.

FILES:
ADDTIME.ARC
ALT-D.ARC
ISMEMO.ARC
PROTO.ARC
REGRESS.ARC

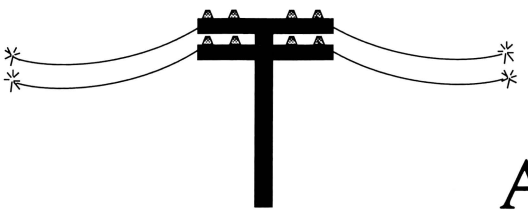


1 2 3 4 5 6

FILES:
ADDTIME.ARC
ALT-D.ARC
ISMEMO.ARC
PROTO.ARC
REGRESS.ARC



A-T BBS



ASHTON·TATE

ELECTRONIC SUPPORT

Your direct line to Ashton-Tate's technical archives begins with a modem and a call to our Support BBS. As a user of this service, you have access to more technical product information than you ever thought possible. Here are a few of the topics we offer:

1. Detailed **usage tips** to help you get the most out of each and every product feature. Things your manual never told you.
2. Descriptions of known **anomalies** ("bugs") and the most effective workarounds. *Knowing* is half the solution.
3. **Articles** which give step-by-step instructions for creating programs, macros, file systems, import/export, theory, and much more. Why reinvent the wheel?
4. New **drivers** for screens, printers, plotters, and keyboards. In minutes, you can download, install, and have running any of these late-breaking drivers. Who needs U.S. Mail?
5. **Sample files** and applications, including simple blocks of exemplary code, application utilities, DOS utilities, function libraries, multi-user programs, templates, and much more.
6. A busy **message board** where you can pose questions to Support Technicians or to other users. There are so many interesting ideas posted on the message board that you will learn something new every time you read it!
7. **Surveys** you can fill out to help us know what you want—now and for the future.
8. **Classified ads** to let other BBS members know about your own products and/or services.

A FREE service...24 hours a day!

Electronic Support is the wave of the future... be a part of it!

(213) 538-6196 (modem)
1200/300 baud, 8–N–1

TechNotes CLASSIFIEDS

All descriptions of services and products have been provided by the advertiser. These products and services have not been tested by Ashton-Tate and their inclusion in this section should not be taken as an endorsement of them by Ashton-Tate.

USER GROUPS

dBug SIG
Dallas, TX
(214) 495-8828

Picture City PC Programming Club
P.O. Box 20342
Rochester, NY 14602

Wiregrass Microcomputer Society
P.O. Box 1082
Enterprise, AL 36331

Hawaii PC User Group
P.O. Box 22967
Honolulu, HI 96822

Capital PC Users Group
Framework SIG
9420 Copenhaver Dr.
Potomac, MD 20854

Central Kentucky Computer Societey, Inc
Suite 100
Security Trust Building
Lexington, KY 40507

Chesapeake PC Users Group, Inc
P.O. Box 2365
Annapolis, MD 21404

dBASE SIG
St. Louis User's Group
for the IBM PC
P.O. Box 1345
St. Louis, MO 63043

dBASE Users Group
Picadilly Cafeteria
Centerville Rd and LBJ
Dallas, TX 73025

Greater South Bay PC Users Group
P.O. Box 1937
Redondo Beach, CA 90278-0937

Manasota IBM PC User Group
4500 Gulf of Mexico Dr. G301
Longboat Key, FL 33548

Space Coast PC Users Group
P.O. Box 396
Titusville, FL 32781-0396

dBASE III Developers
Attn: Gil Orlik
2240 Buchanan, No. 5
San Francisco, CA 94115-9991

Philadelphia Area Computer
Society Framework SIG
2041 Harbour Drive
Palmyra, NJ 08065

Alaska Computer Society
6401 Downey Finch Dr.
Anchorage, AK 99516

San Diego Computer Society
IBM PC SIG
P.O. Box 87770
San Diego, CA 92138

Pemi-Baker Computer Users Group
RFD #2 Box 399
Plymouth, NH 03264

Richmond dBASE Users Group
4206 Seminary Avenue
Richmond, VA 23227

Tampa Database Users Group
119 Plantation Court, Suite D
Tampa, FL 33617-3731

The Boston Computer Society
One Center Plaza
Boston, MA 02108

Garden State Business Users Group
Lackawanna Plaza
Millburn, NJ 07041

Kansas City IBM PC Users Group
221 E. 3rd Street
Kansas City, MO 64113

UCLA PC Users Group
NPI 77-365, 760 Westwood Plaza
Los Angeles, CA 90024

Utah Blue Chips
P.O. Box 510811
Salt Lake City, UT 84151

NYPC dBASE SIG
40 Wall Street, Suite 2124
New York, NY 10005

Terre Haute PC User Group
100 South 22nd Street
Terre Haute, IN 47803

MASTERMENU™ ONLY $37

Instantly create/delete entire menu's
•Built-in menu generator •Passwords
•and more . . .
(207) 439-5074
ATI, Lyndon Way, Kittery, ME, 03904

To receive an information sheet, call 213/538-7573

Rates: $1 per word. Minimum of 50 words. Submit copy and payment. We will reduce/enlarge your art to fit available sizes.

Payment: Full payment must accompany each order. Mail copy and payment to:

TechNotes Classifieds Attn: Jude McGee, 20101 Hamilton Avenue, Torrance, CA 90502-1319

Agency Commission: No agency commission on regular or display classifieds.

Closing Date: Ads must be received 15th of the month two months prior to cover date of magazine (that is, November 15th is the deadline for the January issue).

Frequency discount: 10% discount given for four consecutive insertions

Acceptance: Publisher reserves the right to accept or reject advertisements for Classifieds at their discretion.

Tear Sheets: A tear sheet showing classified ad as printed is sent without charge to each classified advertiser.

Phone orders: Not accepted.

User Groups: Ads are free. Name and address only. Limited to available space.

300 Personals

STRESS ON THE JOB?

Overworked? Burnout?
Depressed? Backache?

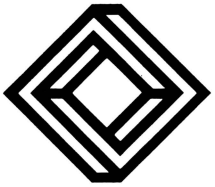
Cheer up! We prescribe Genifer for dBASE programming without tears.

Genifer's the easy to use application generator that creates perfect applications in minutes.

Not only can it save you time, cranking out code. Genifer allows you to sit down with a user, listen to his or her requirements, and bring up a high-quality prototype in under an hour!

Call today for your free demo disk.

Bytel Corporation, 1029 Solano Ave. Berkeley, CA 94706. (415) 527-1157
1-800-631-2229 (CA 1-800-541-3366)



SUBSCRIBE TODAY TO TechNotes AND SAVE 30% OFF THE COVER PRICE !

Get Ashton-Tate's hottest source of technical know-how! Each monthly issue of ***TechNotes*** provides you with the most current, accurate technical information available from the technicians at our Software Support Center on all dBASE, Framework, and RapidFile products. Return this card today! Or subscribe by phone!

Call our toll-free numbers to subscribers: **(800) 331-4164** (in California) or **(800) 828-2514** (residents of other states). In Canada, call **(619) 487-3212**. Cover price is $6.

☐ 1 YEAR for $49.97 Save $22.03 ! ☐ 2 YEARS for $89.97 Save $54.03 ! ☐ 3 YEARS for $124.97 Save $91.03 !

Add $15 for Canada and Mexico. Add $25 for overseas.

☐ Bill me ☐ Check enclosed (Make payable to Ashton-Tate in U.S. dollars drawn on a U.S. bank)

☐ Visa: ☐ MasterCard:

Expiration date: Signature:

☐ New subscription ☐ Renewal

Name

Address

City State/Province Zip Code

Country Phone Number

Which product do you use? ☐ dBASE II (D2) ☐ dBASE III (D3) ☐ dBASE III PLUS (DP) ☐ Framework (FW) ☐ RapidFile (RF) Please allow 30 to 60 days for delivery of your first issue.

1000

READER SURVEY

We would like to make sure that ***TechNotes*** continues to serve the needs of our readership. Please take the time to complete this survey.

Which Ashton-Tate product do you use?

☐ dBASE II ☐ dBASE III/dBASE III PLUS ☐ dBASE III PLUS Lan Pack
☐ Framework II ☐ RapidFile

How long have you been using these product(s)? _____ years _____ months

How long have you been a subscriber to ***TechNotes***? _____

Would you be interested in subscribing to ***TechNotes*** on disk? ☐ Yes ☐ No

Do you download the programs in ***TechNotes*** from the Ashton-Tate SIG on CompuServe or The Source? ☐ Yes ☐ No

Please rate these articles on a scale from 1 to 5 for accuracy, usefulness,and relevancy:

	Accuracy	Usefulness	Relevancy
Format to Program	☐	☐	☐
Detecting Text in Memo Fields	☐	☐	☐
Regression Analysis	☐	☐	☐
Function Prototyping	☐	☐	☐
Soft Record–locking Subroutine	☐	☐	☐

IF YOU'RE MOVING

Don't forget to tell us so that we can be sure your subscription to **TechNotes** and/or the Ashton-Tate Quarterly is uninterrupted. To process a change of address, we must have your previous mailing address (including the zip code) and your new address.

My new address is:

Name/Department

Company Name/Add'l Destination Information

Mailing Address

City/State Zip Country

Please affix your current mailing label(s) here. (If the label is not available, be sure to include your old address.)

Affix your current Ashton-Tate Quarterly mailing label here

Affix your current TechNotes mailing label here

To change your address on the Ashton-Tate Product Mailing List, necessary to receive upgrade and other important information about your Ashton-Tate product(s), you must send a separate change of address to: Ashton-Tate Customer Service, 20101 Hamilton Ave., Torrance, CA 90502-1319.

Tech

All descriptions o
products and serv
should not be tak

USER GR

dBug SIG
Dallas, TX
(214) 495-8828

Picture City PC F
P.O. Box 20342
Rochester, NY 14

Wiregrass Microc
P.O. Box 1082
Enterprise, AL 3(

Hawaii PC User (
P.O. Box 22967
Honolulu, HI 96

Capital PC Users
Framework SIG
9420 Copenhave
Potomac, MD 20

Central Kentucky
Suite 100
Security Trust Bu
Lexington, KY 4(

Chesapeake PC U
P.O. Box 2365
Annapolis, MD 2

dBASE SIG
St. Louis User's (
for the IBM PC
P.O. Box 1345
St. Louis, MO 63

dBASE Users Gr
Picadilly Cafeter
Centerville Rd a
Dallas, TX 7302

Greater South B
P.O. Box 1937
Redondo Beach,

Manasota IBM P
4500 Gulf of Me
Longboat Key, F

Space Coast PC
P.O. Box 396
Titusville, FL 32

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 66 TORRANCE,CA

POSTAGE WILL BE PAID BY ADDRESSEE

TechNotes

Subscription Service
P.O. Box 3729
Escondido, CA 92025-0929

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 66 TORRANCE,CA

POSTAGE WILL BE PAID BY ADDRESSEE

TechNotes

Reader Survey
Ashton-Tate Software Support Center
20101 Hamilton Avenue
Torrance, CA 90502-1319

Attention: Brenda Johnson-Grau

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 66 TORRANCE,CA

POSTAGE WILL BE PAID BY ADDRESSEE

Ashton-Tate Periodicals
PO Box 3729
Escondido, CA 92025-0929

TechNotes
P.O. Box 3729
Escondido, CA 92025-0929

Address Correction Requested

BULK RATE
U.S. POSTAGE
PAID
PERMIT NO. 324
GARDENA, CA